On observing dynamic prioritised actions in SOC*

Rosario Pugliese[1], Francesco Tiezzi[1], and Nobuko Yoshida[2]

[1] Dipartimento di Sistemi e Informatica, Università degli Studi di Firenze
Viale Morgagni 65, 50134 Firenze, Italia
{pugliese,tiezzi}@dsi.unifi.it

[2] Department of Computing, Imperial College London
180 Queen's Gate, South Kensington Campus, SW7 2AZ London, United Kingdom
yoshida@doc.ic.ac.uk

WORKING DRAFT – July 31, 2009

Abstract. In Service-Oriented Computing (SOC), priority mechanisms that combine dynamic priority with local pre-emption enable to model flexible service interactions and fine-grained failure handling. This paper studies the impact of these priority mechanisms on observational semantics for SOC in the process calculus COWS, a sort of extension of asynchronous π-calculus obtained by integrating the above priority mechanisms and other distinctive features of SOC systems, such as, e.g., correlation and pattern matching-based communication, shared variables among parallel threads, activities causing termination. We define natural notions of strong and weak open barbed bisimilarities and prove their coincidence with more manageable characterisations in terms of labelled bisimilarities. These semantics show that COWS's priority mechanisms partially recover the capability to observe receive actions (that could not be observed in a purely asynchronous setting) and that high priority primitives for termination impose specific conditions on the bisimulations. We also show an application of our theories to a 'Morra game' service written in COWS.

* This work has been supported by the EU project SENSORIA, IST-2 005-016004, and EPSRC GR/T03215, GR/T03208 and EP/F003757.

Table of Contents

1 Introduction 3
2 A 'Morra game' scenario 4
3 μCOWS^m : the fragment of μCOWS without priority 7
3.1 Syntax 8
3.2 Operational semantics 9
3.3 Observational semantics 12
Strong open barbed bisimilarity 12
Strong labelled bisimilarity 12
Weak open barbed and labelled bisimilarities 16
4 μCOWS : the protection- and kill-free fragment of COWS 17
4.1 Syntax and operational semantics 17
4.2 Observational semantics 19
Strong open barbed bisimilarity 19
Strong labelled bisimilarity 20
Weak open barbed and labelled bisimilarities 23
5 COWS 23
5.1 Syntax 24
5.2 Operational semantics 24
5.3 Observational semantics 26
Strong open barbed bisimilarity 26
Strong labelled bisimilarity 26
Weak open barbed and labelled bisimilarities 28
5.4 Analysing the 'Morra game' scenario 29
6 Concluding remarks and related work 30
A Proofs of main results 35
A.1 μCOWS^m 35
A.2 μCOWS 40
A.3 COWS 44

1 Introduction

Service-oriented computing (SOC) is an emergent paradigm for distributed computing that aims to build networks of interoperable applications through the use of platform-independent, reusable software components called *services*. Service definitions are used as templates for creating service instances that supply application functionalities to either end-user applications or other instances. Being inherently loosely coupled, SOC systems do not provide intrinsic mechanisms to identify service instances for delivering messages and to link together actions executed as part of the same client-service long-running interaction. Therefore, emerging standards like WS-BPEL and WS-CDL advocate the use of *correlation data* within exchanged messages that the interacting partners can retrieve by means of a *pattern matching* mechanism.

Recently, many process calculi have been expressly designed to model SOC scenarios, so that service definitions and service instances are represented as reactive processes running concurrently. In this setting, priority mechanisms which allow some actions to take precedence over others can be very fruitful. E.g., when a message arrives, the problem arises of rightly handling race conditions among those service instances and the corresponding service definition which are able to receive the message. This can be modelled by exploiting a parallel composition operator that gives precedence to actions with greater priority. Receive activities are then assigned priority values which depend on the messages available so that, in presence of concurrent matching receives, only a receive using a more defined pattern (i.e. having greater priority) can proceed. This way, service instances take precedence over the corresponding service definition when both can process the same message, thus preventing creation of wrong new instances.

Notably, receives would have *dynamically* assigned priority values since these values depend on the matching ability of their argument pattern. Indeed, while computation proceeds, some of the variables used in the argument pattern of a receive can be assigned values, because of execution of syntactically preceding receives or of concurrent threads sharing these variables. This restricts the set of messages matching the pattern while increases the priority of the receive. Furthermore, pre-emption is *local* since receives having a more defined pattern have a higher execution priority with respect to only the other receives matching the same message.

There are other situations where local pre-emption is needed. For example, when a fault arises in a scope, (some of) the remaining activities of the enclosing scope should be terminated before starting the execution of the relative fault handler. This can be modelled by exploiting the same parallel operator as before together with actions for forcing immediate termination of concurrent activities which take the greatest priority. The same mechanism can also be used for exception and compensation handling.

However, apart from COWS [33, 32], priority mechanisms that combine dynamic priority with local pre-emption have not been studied yet in the literature [16]. COWS (*Calculus for Orchestration of Web Services*) is an extension of asynchronous π-calculus [28, 3] equipped with the priority mechanisms sketched above and with other distinctive features of SOC systems inspired by WS-BPEL, such as shared variables among parallel threads, and communication based on correlation and pattern matching.

This paper studies the impact of COWS's priority mechanisms on observational semantics for SOC. The obtained semantic theories are directly usable to check inter-

changeability of services and conformance against service specifications. They can also be used to reduce the size of the model representing services, thus e.g. facilitating model checking of service properties [19]. Since we want to investigate how COWS distinctive features affect well-established semantic theories, we present syntax, operational and observational semantics of COWS in three steps. This will also allows us to gradually introduce the technicalities of our development. Thus, in Section 3 we consider μCOWS^m (μCOWS *minus priority*), a fragment of COWS without priority in parallel composition and linguistic constructs dealing with termination. It retains all the other COWS's features, like e.g. global scope and pattern matching. In Section 4 we move on μCOWS (*micro* COWS), the calculus obtained by enriching μCOWS^m with priority in parallel composition. Finally, in Section 5 we study the full calculus, that extends μCOWS with primitives for termination.

The paper contains four main contributions.

1. In Section 3, to understand the effect of non-binding and localised receives, pattern-matching, and global scope on the semantics, we first define strong and weak open barbed bisimilarities [43, 29] for μCOWS^m. We then provide more manageable labelled bisimilarities and prove that they are *sound* and *complete* with respect to barbed (contextual) ones. Both semantics recall those for asynchronous π-calculus of [3]. A major complication is that, due to the locality of received endpoints (i.e. only the output capability of names may be transmitted, as in the *localised π-calculus* [34]), the definition of labelled bisimilarities involves a family of relations indexed by sets of names. This is somewhat similar to the definition of quasi-open bisimilarity for π-calculus [43].
2. In Section 4, to understand the effect of actions with dynamic changing priority on the semantics, we move on μCOWS. Again, we provide sound and complete characterisations of the barbed bisimilarities in terms of corresponding labelled bisimilarities. The obtained semantics inhabits between asynchrony and synchrony because, with respect to a purely asynchronous setting, the priority mechanism permits partially recovering the capability to observe receive actions.
3. In Section 5, we extend our investigation to COWS. The primitives with greatest priority causing termination require specific conditions on the labelled bisimilarities for these to be congruences. The resulting observations are hence more fine-grained than the previous ones. Results of coincidence still hold.
4. In Sections 2 and 5, we show a use of the semantic theories developed through a practical example of web service which includes all key features of COWS.

Section 6 concludes the paper with a presentation of related and future work, while Appendix A reports a full account of the proofs.

2 A 'Morra game' scenario

Before formally defining COWS, we provide some insights into its main features in a step-by-step fashion by means of an example. This is a service inspired by the well-known game Morra[1] and described at two different levels of abstraction.

[1] For further information about the Morra game visit the web site `http://en.wikipedia.org/wiki/Morra_(game)`.

Let us consider a service that allows its clients to play the Morra game. We consider the variation of Morra where two players, named "odds" and "evens", throw out a single hand, each showing zero to five fingers. If the sum of fingers shown by both players is an even number then the "evens" player wins; otherwise the "odds" player is the winner. The service collects the two throws (i.e. two integers), calculates the winner and sends the result back to the two players. A high-level specification of the service in COWS is:

$$\begin{array}{l} * \, [x_{id}, x_p, x_{num}, y_p, y_{num}] \\ \quad (\mathit{odds} \cdot \mathit{throw}?\langle x_{id}, x_p, x_{num}\rangle \mid \mathit{evens} \cdot \mathit{throw}?\langle x_{id}, y_p, y_{num}\rangle \\ \qquad \mid x_p \cdot \mathit{res}!\langle x_{id}, \mathit{win}(x_{num}, y_{num}, 1)\rangle \mid y_p \cdot \mathit{res}!\langle x_{id}, \mathit{win}(x_{num}, y_{num}, 0)\rangle \,) \end{array} \tag{1}$$

The service receives throws from the players via two distinct endpoints, i.e. pairs $\mathit{odds} \cdot \mathit{throw}$ and $\mathit{evens} \cdot \mathit{throw}$, which can be interpreted as specific implementations of the *operation* name *throw* provided by *partner* names *odds* and *evens*. The players are required to provide the partner names, stored in variables x_p and y_p, that they will use to receive the result. The replication operator $*\_$, that spawns in parallel as many copies of its argument term as necessary, permits supporting creation of multiple instances to serve several matches simultaneously. A match is identified by a match-id, stored in x_{id}, that the partners need to provide when sending their throws. To avoid interferences between matches played simultaneously, match-ids should be unique (clients can use the delimitation operator $[\_]$ to guarantee uniqueness). Partner throws arrive randomly, thus any interaction with the service starts with one of the two *receive* activities $\mathit{odds} \cdot \mathit{throw}?\langle x_{id}, x_p, x_{num}\rangle$ or $\mathit{evens} \cdot \mathit{throw}?\langle x_{id}, y_p, y_{num}\rangle$, corresponding to reception of throws of the match identified by x_{id}, and terminates with the two *invoke* activities $x_p \cdot \mathit{res}!\langle x_{id,}\mathit{win}(x_{num}, y_{num}, 1)\rangle$ and $y_p \cdot \mathit{res}!\langle x_{id,}\mathit{win}(x_{num}, y_{num}, 0)\rangle$, used to reply with the result. We assume that $\mathit{win}(x, y, z)$ is a total function which, if x and y are integers between 0 and 5, returns the string w (abbreviation of 'winner') in case $(x + y)\, \mathit{mod}\ 2$ is equal to z, and the string l (abbreviation of 'loser') if $(x + y)\, \mathit{mod}\ 2$ is different from z; otherwise, the string *err* is returned.

Service (1) uses the delimitation operator to declare the scope of variables x_{id}, x_p, y_p, x_{num} and y_{num}. An inter-service communication takes place when the arguments of a receive and of a concurrent invoke along the same endpoint match and causes replacement of the variables arguments of the receive with the corresponding values arguments of the invoke (within the scope of variables declarations). Notably, the two receive activities are *correlated* by means of the *shared* variable x_{id}.

When an invocation for operation *throw* is processed, it must be checked if a service instance with the same match-id already exists, in which case the invocation is received by the instance, or if the service must produce a new instance. This is done through the *dynamic prioritised mechanism* of COWS, i.e. assigning the receives by instances (having a more defined pattern) a greater priority than the receives by the service definition.

Thus, for example, after an interaction with the following client

$$[z]\,(\, \mathit{evens} \cdot \mathit{throw}!\langle \mathit{first}, \mathit{cbB}, 1\rangle \mid \mathit{cbB} \cdot \mathit{res}?\langle \mathit{first}, z\rangle \,.\, \langle \text{rest of client B}\rangle \,)$$

service definition (1) runs in parallel with the instance identified by the match-id *first* (the instance is highlighted by a gray background)

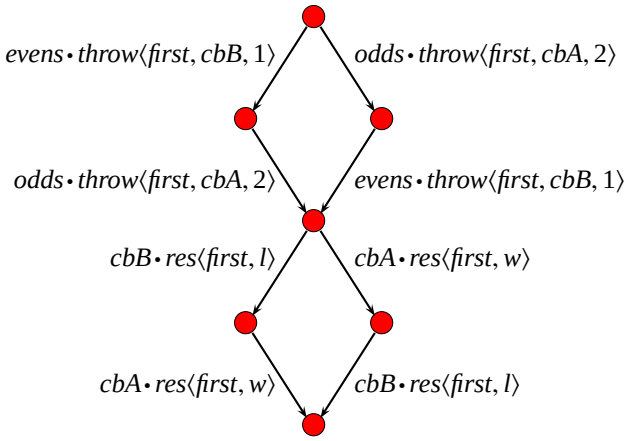

Fig. 1. Long-running interaction between clients and high-level Morra specification

$$
\begin{aligned}
&*\,[x_{id}, x_p, x_{num}, y_p, y_{num}] \\
&\quad (\, odds \cdot throw?\langle x_{id}, x_p, x_{num}\rangle \mid evens \cdot throw?\langle x_{id}, y_p, y_{num}\rangle \\
&\qquad \mid x_p \cdot res!\langle x_{id}, win(x_{num}, y_{num}, 1)\rangle \mid y_p \cdot res!\langle x_{id}, win(x_{num}, y_{num}, 0)\rangle\,) \\
&\mid\ [x_p, x_{num}]\ (\, odds \cdot throw?\langle first, x_p, x_{num}\rangle \\
&\qquad\qquad \mid x_p \cdot res!\langle first, win(x_{num}, 1, 1)\rangle \mid cbB \cdot res!\langle first, win(x_{num}, 1, 0)\rangle\,)
\end{aligned}
$$

Now, if another client performs the invocation $odds \cdot throw!\langle first, cbA, 2\rangle$, it will be processed by the already existing instance because, w.r.t. this invocation, the receive $odds \cdot throw?\langle first, x_p, x_{num}\rangle$ has greater priority than the receive $odds \cdot throw?\langle x_{id}, x_p, x_{num}\rangle$ (that has a less defined argument pattern). The long-running interaction between the Morra service definition (1) and the above clients is graphically represented in Figure 1, where each node represents a state, while each edge describes a communication action (e.g. label $evens \cdot throw\langle first, cbB, 1\rangle$ denotes taking place of a communication along endpoint $evens \cdot throw$ with matching values $\langle first, cbB, 1\rangle$).

For a lower level implementation, we wish to maximise the abilities of different services, while preserving the observable behaviour of the whole service w.r.t. the high-level specification. The main service is now composed of three entities as follows:

$$[req2f, req5f, resp2f, resp5f]\,(\,*\,M \mid *\,2F \mid *\,5F\,) \tag{2}$$

The delimitation operator is used here to declare that $req2f$, $req5f$, $resp2f$ and $resp5f$ are private operation names known to the three components M, $2F$ and $5F$, and only to them. The three subservices are defined as follows:

$$
\begin{aligned}
M \triangleq\ & [x_{id}, x_p, x_{num}, y_p, y_{num}] \\
& (\, odds \cdot throw?\langle x_{id}, x_p, x_{num}\rangle \mid evens \cdot throw?\langle x_{id}, y_p, y_{num}\rangle \\
& \quad \mid [k]\,(\, m \cdot req2f!\langle x_{id}, x_{num}, y_{num}\rangle \mid m \cdot req5f!\langle x_{id}, x_{num}, y_{num}\rangle \\
& \qquad \mid [x_o, x_e]\, m \cdot resp2f?\langle x_{id}, x_o, x_e\rangle. \\
& \qquad\qquad (\,\mathbf{kill}(k) \mid \{\!|x_p \cdot res!\langle x_{id}, x_o\rangle \mid y_p \cdot res!\langle x_{id}, x_e\rangle|\!\}\,) \\
& \qquad \mid [x_o, x_e]\, m \cdot resp5f?\langle x_{id}, x_o, x_e\rangle. \\
& \qquad\qquad (\,\mathbf{kill}(k) \mid \{\!|x_p \cdot res!\langle x_{id}, x_o\rangle \mid y_p \cdot res!\langle x_{id}, x_e\rangle|\!\}\,)\,)
\end{aligned}
$$

$$
\begin{aligned}
2F \triangleq\ & [x]\,(\, m \bullet req2f?\langle x,1,1\rangle.\, m \bullet resp2f!\langle x,l,w\rangle \\
& \quad + m \bullet req2f?\langle x,1,2\rangle.\, m \bullet resp2f!\langle x,w,l\rangle \\
& \quad + m \bullet req2f?\langle x,2,1\rangle.\, m \bullet resp2f!\langle x,w,l\rangle \\
& \quad + m \bullet req2f?\langle x,2,2\rangle.\, m \bullet resp2f!\langle x,l,w\rangle\,)
\end{aligned}
$$

$$
\begin{aligned}
5F \triangleq\ & [x,y,z]\,(\, m \bullet req5f?\langle x,y,z\rangle.\, m \bullet resp5f!\langle x,err,err\rangle \\
& \quad + m \bullet req5f?\langle x,0,0\rangle.\, m \bullet resp5f!\langle x,l,w\rangle \\
& \quad + m \bullet req5f?\langle x,0,1\rangle.\, m \bullet resp5f!\langle x,w,l\rangle \\
& \quad + \ldots + m \bullet req5f?\langle x,5,5\rangle.\, m \bullet resp5f!\langle x,l,w\rangle\,)
\end{aligned}
$$

Service M is publicly invocable and can interact with players as well as with the 'internal' services $2F$ and $5F$. These latter two services, instead, can only be invoked by M and have the task of calculating the winner of a match. In particular, $2F$ performs a quick computation of simple matches where both players hold out either one or two fingers, while $5F$ performs a slower computation of standard 5-fingers matches (that exactly corresponds to the computation modelled by the function $win(\_)$). After the two initial receives, for e.g. performance and fault tolerance purposes, M invokes services $2F$ and $5F$ concurrently. Communication between M and the other two subservices relies on the match identifier (stored in x) as correlation data. When one of $2F$ and $5F$ replies, M immediately stops the other computation. This is done by executing the *kill* activity **kill**(k), that forces termination of all unprotected parallel terms inside the enclosing $[k]$, which stops the killing effect. Kill activities are executed *eagerly* w.r.t. the other parallel activities included within the enclosing scope, because relatively to these latter activities they take the greatest priority. However, critical activities can be protected from the effect of a forced termination by using the *protection* operator $\{\!|\_|\!\}$; this is indeed the case of the response $x_p \bullet res!\langle x_{id}, x_o\rangle$ in our example. Finally, M forwards the responses to the players and terminates.

Services $2F$ and $5F$ use the *choice* operator $\_ + \_$ to offer alternative behaviours: one of them can be selected by executing an invoke matching the receive leading the behaviour. If the throws are not integers between 0 and 5, $2F$ does not reply, while $5F$ returns the string *err*. Indeed, the receive $m \bullet req5f?\langle x,y,z\rangle$ is assigned less priority than the other receive activities, i.e. it is only executed when none of the other receives matches the two throws, thus avoiding to return *err* in case of admissible throws.

An interaction of specification (2) with the previous clients is shown in Figure 2, where $\emptyset(m \bullet o)$ represents an internal communication along the endpoint $m \bullet o$, while $\dagger(m \bullet o?\bar{w})$ denotes an invisible kill-action that terminates the activity $m \bullet o?\bar{w}$ (and its continuation). Dotted lines stand for alternative behaviours. Notably, after execution of a communication along $m \bullet resp2f$, the prioritised semantics of COWS permits executing only the kill action.

In the sequel, we will show how dynamic priorities with local pre-emption and (invisible) kill activities affect the observable behaviours. In Section 5, we will study conformance between specifications (1) and (2) using a weak bisimilarity for COWS.

3 μCOWS^m : the fragment of μCOWS without priority

The fragment of COWS introduced in this section, namely μCOWS^m, dispenses with priority in parallel composition and linguistic constructs dealing with termination.

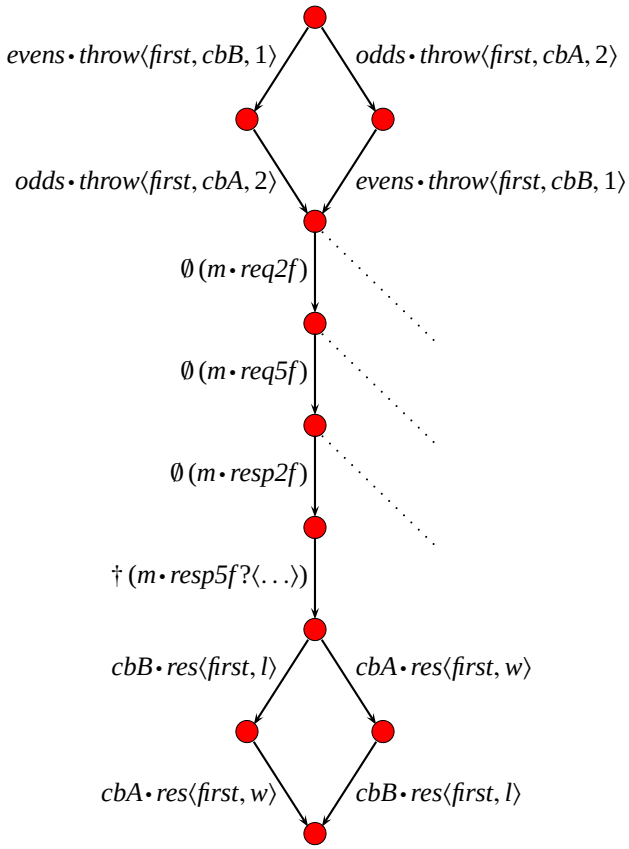

Fig. 2. Long-running interaction between clients and low-level Morra specification

3.1 Syntax

The syntax of μCOWS^m is presented in Table 1. We use two countable and disjoint sets: the set of *values* (ranged over by v, v', ...) and the set of 'write once' *variables* (ranged over by x, y, ...). The set of values is left unspecified; however, we assume that it includes the set of *names* (ranged over by n, m, p, o, ...) mainly used to represent partners and operations. We also use a set of *expressions* (ranged over by ϵ), whose exact syntax is deliberately omitted; we just assume that expressions contain, at least, values and variables. Partner names and operation names can be combined to designate *endpoints*, written $p \cdot o$, and can be communicated, but dynamically received names can only be used for service invocation (as in *localised π-calculus* [34]). Indeed, endpoints of receive activities are identified statically because their syntax only allows using names and not variables.

In the sequel, w ranges over values and variables and u ranges over names and variables. Notation $\bar{\cdot}$ stands for tuples, e.g. $\bar{x}$ means $\langle x_1, \ldots, x_n \rangle$ (with $n \geq 0$) where variables in the same tuple are pairwise distinct. We write $a, \bar{b}$ to denote the tuple obtained by concatenating the element a to the tuple $\bar{b}$. All notations shall extend to tuples component-wise. $\mathtt{n}$ ranges over communication endpoints that do not contain variables

s	::=		(services)	g	::=		(receive-guarded choice)
		$u \bullet u'!\bar{\epsilon}$	(invoke)			$\mathbf{0}$	(nil)
	$\mid$	g	(receive-guarded choice)		$\mid$	$p \bullet o?\bar{w}.s$	(request processing)
	$\mid$	$s \mid s$	(parallel composition)		$\mid$	$g + g$	(choice)
	$\mid$	$[u]\, s$	(delimitation)				
	$\mid$	$* s$	(replication)				

Table 1. μCOWS^m syntax

(e.g. $p \bullet o$), while $\mathtt{u}$ ranges over communication endpoints that may contain variables (e.g. $u \bullet u'$). Sometimes, we will use notation $\mathtt{n}$ and $\mathtt{u}$ for the tuples $\langle p, o\rangle$ and $\langle u, u'\rangle$, respectively, and rely on the context to resolve any ambiguity. When convenient, we shall regard a tuple (hence, also an endpoint) simply as a set, writing e.g. $x \in \bar{y}$ to mean that x is an element of $\bar{y}$. We will omit trailing occurrences of $\mathbf{0}$, writing e.g. $p \bullet o?\bar{w}$ instead of $p \bullet o?\bar{w}.\mathbf{0}$, and write $[\langle u_1, \ldots, u_n\rangle]\, s$ in place of $[u_1] \ldots [u_n]\, s$. We will write $I \triangleq s$ to assign a name I to the term s.

We adopt the following conventions about the operators precedence: monadic operators bind more tightly than parallel composition, and prefixing more tightly than choice.

The only *binding* construct is delimitation: $[u]\, s$ binds u in the scope s. In fact, to enable concurrent threads within each service instance to share (part of) the state, receive activities in COWS bind neither names nor variables. The occurrence of a name/variable is *free* if it is not under the scope of a delimitation for it. Bound and free names are also called *private* and *public* names, respectively. We denote by $\mathrm{fu}(t)$ the set of free names/variables that occur free in t. Two terms are *α-equivalent* if one can be obtained from the other by consistently renaming bound names/variables. As usual, we identify terms up to α-equivalence.

3.2 Operational semantics

The operational semantics of μCOWS^m is defined only for *closed* services, i.e. services without free variables. Formally, the semantics is given in terms of a structural congruence and of a labelled transition relation. The *structural congruence*, written $\equiv$, identifies syntactically different services that intuitively represent the same service. It is defined as the least congruence relation induced by the equational laws shown in Table 2. All the laws are straightforward. In particular, commutativity of consecutive delimitations implies that the order among the u_i in $[\langle u_1, \ldots, u_n\rangle]\, s$ is irrelevant, thus in the sequel we may use the simpler notation $[u_1, \ldots, u_n]\, s$. The last law permits to extend the scope of names (as in the π-calculus) and variables, thus enabling possible communication.

To define the labelled transition relation, we use two auxiliary functions. Firstly, we use the function $[\![\_]\!]$ for evaluating *closed* expressions (i.e. expressions without variables): it takes a closed expression and returns a value. It is not explicitly defined since the exact syntax of expressions is deliberately not specified. Secondly, we use the partial function $\mathcal{M}(\_, \_)$ for performing *pattern-matching* on semi-structured data and, thus, determining if a receive and an invoke over the same endpoint can synchronise. The rules

$$
\begin{array}{rcl@{\qquad}rcl@{\qquad}rcl}
*\mathbf{0} & \equiv & \mathbf{0} & *s & \equiv & s \mid *s & & & \\
s \mid \mathbf{0} & \equiv & s & s_1 \mid s_2 & \equiv & s_2 \mid s_1 & (s_1 \mid s_2) \mid s_3 & \equiv & s_1 \mid (s_2 \mid s_3) \\
g + \mathbf{0} & \equiv & g & g_1 + g_2 & \equiv & g_2 + g_1 & (g_1 + g_2) + g_3 & \equiv & g_1 + (g_2 + g_3) \\
[u]\,\mathbf{0} & \equiv & \mathbf{0} & [u_1]\,[u_2]\,s & \equiv & [u_2]\,[u_1]\,s & s_1 \mid [u]\,s_2 & \equiv & [u]\,(s_1 \mid s_2) \quad \text{if } u \notin \mathrm{fu}(s_1)
\end{array}
$$

Table 2. μCOWS^m structural congruence

$$
\mathcal{M}(x, v) = \{x \mapsto v\} \qquad \mathcal{M}(v, v) = \emptyset \qquad \mathcal{M}(\langle\rangle, \langle\rangle) = \emptyset \qquad \frac{\mathcal{M}(w_1, v_1) = \sigma_1 \qquad \mathcal{M}(\bar{w}_2, \bar{v}_2) = \sigma_2}{\mathcal{M}((w_1, \bar{w}_2), (v_1, \bar{v}_2)) = \sigma_1 \uplus \sigma_2}
$$

Table 3. Matching rules

defining $\mathcal{M}(\_, \_)$ are shown in Table 3. They state that two tuples match if they have the same number of fields and corresponding fields have matching values/variables. Variables match any value, and two values match only if they are identical. When tuples $\bar{w}$ and $\bar{v}$ do match, $\mathcal{M}(\bar{w}, \bar{v})$ returns a substitution for the variables in $\bar{w}$; otherwise, it is undefined. *Substitutions* (ranged over by σ) are functions mapping variables to values and are written as collections of pairs of the form $x \mapsto v$. Application of substitution σ to s, written $s \cdot \sigma$, has the effect of replacing every free occurrence of x in s with v, for each $x \mapsto v \in \sigma$, by possibly using α-conversion for avoiding v to be captured by name delimitations within s. We use $\emptyset$ to denote the empty substitution, $|\sigma|$ to denote the number of pairs in σ, and $\sigma_1 \uplus \sigma_2$ to denote the union of σ_1 and σ_2 when they have disjoint domains.

The *labelled transition relation* $\xrightarrow{\alpha}$ is the least relation over services induced by the rules in Table 4, where label α is generated by the following grammar:

$$
\alpha \quad ::= \quad \mathtt{n} \lhd [\bar{n}]\,\bar{v} \quad \mid \quad \mathtt{n} \rhd [\bar{x}]\,\bar{w} \quad \mid \quad \sigma
$$

The meaning of labels is as follows: $\mathtt{n} \lhd [\bar{n}]\,\bar{v}$ and $\mathtt{n} \rhd [\bar{x}]\,\bar{w}$ denote execution of invoke and receive activities over the endpoint $\mathtt{n}$ with arguments $\bar{v}$ and $\bar{w}$, and delimited arguments $\bar{n}$ and $\bar{x}$, respectively; σ denotes execution of a communication with generated substitution σ to be still applied. $\emptyset$ denotes a *computational step* corresponding to taking place of communication without pending substitutions. We write $\mathtt{n} \lhd \bar{v}$ (resp. $\mathtt{n} \rhd \bar{w}$) instead of $\mathtt{n} \lhd [\,]\,\bar{v}$ (resp. $\mathtt{n} \rhd [\,]\,\bar{w}$) and use $\mathrm{u}(\alpha)$ to denote the set of names and variables occurring in α, where $\mathrm{u}(\{x \mapsto v\}) = \{x\} \cup \mathrm{fu}(v)$ and $\mathrm{u}(\sigma_1 \uplus \sigma_2) = \mathrm{u}(\sigma_1) \cup \mathrm{u}(\sigma_2)$. Moreover, we will use $\mathrm{bu}(\alpha)$ to denote the set of names/variables that occur bound in α; i.e. $\mathrm{bu}(\mathtt{n} \lhd [\bar{n}]\,\bar{v}) = \bar{n}$, $\mathrm{bu}(\mathtt{n} \rhd [\bar{x}]\,\bar{w}) = \bar{x}$ and $\mathrm{bu}(\sigma) = \emptyset$.

We comment on salient points. A service invocation can proceed only if the expressions in the argument can be evaluated (rule *(inv)*). This means, for example, that if it contains a variable x it is stuck until x is not replaced by a value because of execution of a receive assigning a value to x. A receive activity offers an invocable operation along a given partner name (rule *(rec)*), and execution of a receive permits to take a decision between alternative behaviours (rule *(choice)*). Bound invocations, which transmit private names, can be generated by rule *(open$_{inv}$)*, while delimited receive activities can proceed

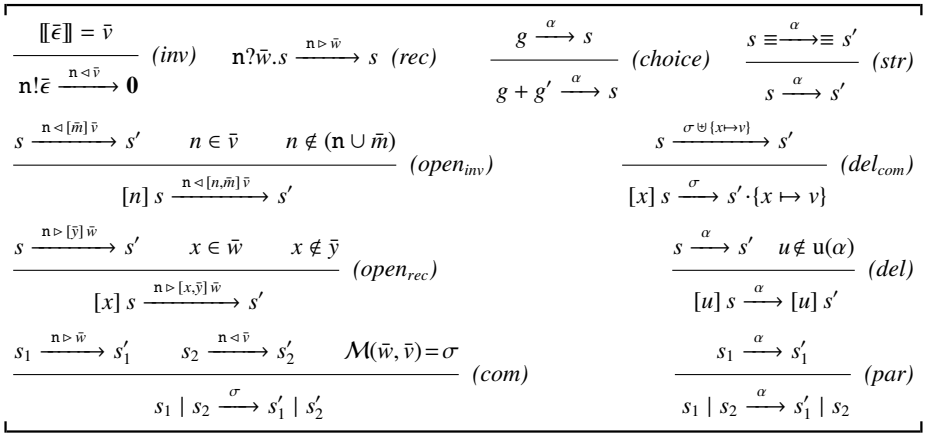

Table 4. μCOWS^m operational semantics

thanks to rule *(open$_{rec}$)*. Communication can take place when two parallel services perform matching receive and invoke activities (rule *(com)*). Communication generates a substitution that is recorded in the transition label (for subsequent application), rather than a silent transition as in most process calculi. When the delimitation of a variable x argument of a receive involved in a communication is encountered, i.e. the whole scope of the variable is determined, the delimitation is removed and the substitution for x is applied to the term (rule *(del$_{com}$)*). Variable x disappears from the term and cannot be reassigned a value (for this reason we say that COWS's variables are 'write once'). $[u]\,s$ behaves like s (rule *(del)*), except when the transition label α contains u. Execution of parallel services is interleaved (rule *(par)*). Rule *(str)* is standard and states that structurally congruent services have the same transitions. Notably, bound invocation actions do not appear in rule *(com)*, and therefore cannot directly interact with receive actions, and similarly delimited receive actions cannot synchronise with invoke actions. Such interactions are instead inferred by using structural congruence to pull name/variable delimitations outside of both interacting activities. Since rules *(open$_{inv}$)* and *(open$_{rec}$)* have no impact on inferring computational steps, they could be omitted from the rules defining the operational semantics. However, when it comes to developing behavioural equivalences, such rules turn out to be indispensable. It is also worth noticing that when rule *(open$_{rec}$)* is applied to a closed μCOWS^m service, the resulting term could be open.

We end with three properties of the operational semantics that will be exploited in the rest of the paper.

Property 1. Let s be a μCOWS^m term. The following facts hold:

- If $s \xrightarrow{\mathrm{n} \lhd [\bar{n}]\,\bar{v}} s'$, then $\bar{n} \subseteq \bar{v}$ and $\mathrm{n} \notin \bar{n}$.
- If $s \xrightarrow{\mathrm{n} \rhd [\bar{x}]\,\bar{w}} s'$, then $\bar{x} \subseteq \bar{w}$. Moreover, if s is closed then $\bar{w} \backslash \bar{x}$ does not contain variables (i.e. it is either a tuple of values or the empty tuple).
- If s is closed and $s \xrightarrow{\sigma} s'$, then $\sigma = \emptyset$ and s' is closed.

The above properties can be proved by a straightforward induction on the depth of the shortest inference for the transitions in the hypothesis.

3.3 Observational semantics

We introduce now an observational semantics for μCOWS^m. Specifically, we define natural notions of strong and weak open barbed bisimilarities and prove their coincidence with more manageable characterisations in terms of labelled bisimilarities.

Strong open barbed bisimilarity. We want to define a notion of *(open) barbed bisimilarity* for the calculus along the line of [29, 43]. To this aim, we must first identify an appropriate *basic observable*, namely a predicate that points out the interaction capabilities of a term. Since communication is asynchronous, an obvious starting point is considering as observable only the output capabilities of terms, like for asynchronous π-calculus [3]. The intuition is that an asynchronous observer cannot directly observe the receipt of data that he has sent. Thus, our notion of observation is as follows.

Definition 1 (Observable for μCOWS^m). *Let s be a μCOWS^m closed term. Predicate $s \downarrow_{\mathtt{n}}$ holds true if s can immediately perform an invoke over the (public) endpoint $\mathtt{n}$, that is if there exist s', $\bar{n}$ and $\bar{v}$ such that $s \xrightarrow{\mathtt{n} \lhd [\bar{n}]\,\bar{v}} s'$.*

A desirable property of a behavioural equivalence, which enables compositional reasoning, is to be preserved by all contexts of the language. An equivalence that enjoys this property is called *congruence*. A μCOWS^m (closed) context is a service $\mathbb{C}$ with a 'hole' $[\![\cdot]\!]$, i.e. a term generated by the following grammar:

$$\mathbb{C} ::= [\![\cdot]\!] \mid \mathbb{G} \mid \mathbb{C} \mid s \mid s \mid \mathbb{C} \mid [u]\,\mathbb{C} \mid * \mathbb{C} \qquad \mathbb{G} ::= \mathtt{n}?\bar{w}.\,\mathbb{C} \mid \mathbb{G} + g \mid g + \mathbb{G}$$

such that, once the hole is filled with a closed service s, the resulting term $\mathbb{C}[\![s]\!]$ is a μCOWS^m closed service.

The above definitions of observation and context lead to the following notion of barbed bisimilarity.

Definition 2 (Open barbed bisimilarity). *A symmetric binary relation $\mathcal{R}$ on μCOWS^m closed terms is an* open barbed bisimulation *if whenever $s_1 \mathcal{R} s_2$ the following holds:*

(Barb preservation) *if $s_1 \downarrow_{\mathtt{n}}$ then $s_2 \downarrow_{\mathtt{n}}$;*

(Computation closure) *if $s_1 \xrightarrow{\emptyset} s_1'$ then there exists s_2' such that $s_2 \xrightarrow{\emptyset} s_2'$ and $s_1' \mathcal{R} s_2'$;*

(Context closure) $\mathbb{C}[\![s_1]\!]\,\mathcal{R}\,\mathbb{C}[\![s_2]\!]$*, for every closed context $\mathbb{C}$.*

Two closed terms s_1 and s_2 are open barbed bisimilar*, written $s_1 \simeq_m s_2$, if $s_1 \mathcal{R} s_2$ for some open barbed bisimulation $\mathcal{R}$. $\simeq_m$ is called open barbed bisimilarity.*

Of course, because of context closure, $\simeq_m$ is a congruence for μCOWS^m closed terms.

Strong labelled bisimilarity. The definition of $\simeq_m$ suffers from universal quantification over all possible language contexts, which makes the reasoning on terms very hard. Hence, we provide a purely co-inductive notion of bisimulation that only requires considering transitions of the labelled transition system defining the semantics of the

terms under analysis. The notion of labelled bisimulation introduced for asynchronous π-calculus in [3] does not turn out to be suitable for μCOWS^m, since the bisimilarity defined on top of it would not properly characterise $\simeq_m$. Consider, for example, the following two μCOWS^m terms:

$$s_1 \triangleq [\mathtt{n}]\,(\,\mathtt{m}!\langle\mathtt{n}\rangle \mid \mathtt{n}!\langle\rangle\,) \qquad\qquad s_2 \triangleq [\mathtt{n}]\;\mathtt{m}!\langle\mathtt{n}\rangle$$

They only differ for the fact that the first one is able to perform an invocation along the private endpoint $\mathtt{n}$. However, they can exhibit the same barbs, and no context can tell them apart since it cannot be able to perform a receive along (the received name) $\mathtt{n}$ because of the constraint on the 'localisation' of names (indeed, contexts of the form $[\![\cdot]\!] \mid \mathtt{m}?\langle x\rangle.\, x?\langle\rangle.\, \mathbf{0}$ are not allowed). Hence, s_1 and s_2 are barbed bisimilar. The natural asynchronous labelled bisimilarity derived from the π-calculus one would instead tell them apart and, hence, need to be weakened. Therefore we define a labelled bisimulation as a family of relations indexed with sets of names corresponding to the names that cannot be used by contexts (to test) for reception since they are dynamically exported private names.

Definition 3 (Names-indexed family of relations). *A names-indexed family $\mathcal{F}$ of relations is a set of symmetric binary relations $\mathcal{R}_\mathcal{N}$ on μCOWS^m closed terms, one for each set of names $\mathcal{N}$, i.e. $\mathcal{F} = \{\mathcal{R}_\mathcal{N}\}_\mathcal{N}$.*

Definition 4 (Labelled bisimilarity). *A names-indexed family of relations $\{\mathcal{R}_\mathcal{N}\}_\mathcal{N}$ is a* labelled bisimulation *if, whenever $s_1 \mathcal{R}_\mathcal{N} s_2$ and $s_1 \xrightarrow{\alpha} s_1'$, where* $\mathrm{bu}(\alpha)$ *are fresh, then:*

1. *if $\alpha = \mathtt{n} \rhd [\bar{x}]\,\bar{w}$ then one of the following holds:*
   (a) $\exists\, s_2' \;:\; s_2 \xrightarrow{\mathtt{n} \rhd [\bar{x}]\,\bar{w}} s_2'$ *and* $\forall\, \bar{v}$ *s.t.* $\mathcal{M}(\bar{x}, \bar{v}) = \sigma \;:\; s_1' \cdot \sigma\, \mathcal{R}_\mathcal{N}\, s_2' \cdot \sigma$
   (b) $\exists\, s_2' \;:\; s_2 \xrightarrow{\emptyset} s_2'$ *and* $\forall\, \bar{v}$ *s.t.* $\mathcal{M}(\bar{x}, \bar{v}) = \sigma \;:\; s_1' \cdot \sigma\, \mathcal{R}_\mathcal{N}\, (s_2' \mid \mathtt{n}!(\bar{w} \cdot \sigma))$
2. *if $\alpha = \mathtt{n} \lhd [\bar{n}]\,\bar{v}$ where $\mathtt{n} \notin \mathcal{N}$ then $\exists\, s_2' \;:\; s_2 \xrightarrow{\mathtt{n} \lhd [\bar{n}]\,\bar{v}} s_2'$ and $s_1'\, \mathcal{R}_{\mathcal{N} \cup \bar{n}}\, s_2'$*
3. *if $\alpha = \emptyset$ then $\exists\, s_2' \;:\; s_2 \xrightarrow{\emptyset} s_2'$ and $s_1'\, \mathcal{R}_\mathcal{N}\, s_2'$*

*Two closed terms s_1 and s_2 are $\mathcal{N}$-*bisimilar*, written $s_1 \sim_m^\mathcal{N} s_2$, if $s_1 \mathcal{R}_\mathcal{N} s_2$ for some $\mathcal{R}_\mathcal{N}$ in a labelled bisimulation. They are* labelled bisimilar*, written $s_1 \sim_m s_2$, if they are $\emptyset$-bisimilar. $\sim_m^\mathcal{N}$ is called $\mathcal{N}$-bisimilarity, while $\sim_m$ is called labelled bisimilarity.*

The resulting definition somewhat recalls that of quasi-open bisimilarity for π-calculus [43]. Clause 1 states that a receive action can be simulated either in a normal way (clause 1.(a)) or by a computational step leading to a term that, when composed with the invoke activity consumed by the receive, stands in the appropriate relation (clause 1.(b)). Execution of receives whose argument contains variables leads to open terms, which the operational semantics is not defined for. Since the freed variables are placeholders for values to be received, we require the two continuations to be related for any matching tuple of values (similarly to late bisimulation for π-calculus [37]). We say that the bisimulation is given in a *late* style because in clause 1 the choice of the tuple of values $\bar{v}$ takes place after the choice of s_2'; that is, s_2' does not depend on the tuple of values $\bar{v}$. Clause 2, and the use of names-indexed families of relations, handles

the fact that dynamically exported private names cannot be used by a receiver within the endpoint of a receive (whose syntax, as we have seen in Section 3.1, does not allow to use variables). With abuse of notation, $\mathtt{n} \notin \mathcal{N}$ in clause 2, with $\mathtt{n} = p \bullet o$, stands for $p \notin \mathcal{N} \wedge o \notin \mathcal{N}$. Thus, invocations along endpoints using either of the names in $\mathcal{N}$ are unobservable, hence these endpoints cannot be used to tell the executing terms apart. Finally, clause 3 deals with computational steps. Notably, actions σ different from $\emptyset$ are not taken into account, since they cannot be performed by closed terms (see rules *(com)* and *(del_{com})*).

To illustrate our labelled bisimilarity, let us consider a tailored version of the input absorption law characterizing asynchronous bisimulation in asynchronous π-calculus (i.e. the equation $a(b).\,\bar{a}b + \tau = \tau$ presented in [3]):

$$[x]\,(\,\emptyset + \mathtt{n}?\langle x, v\rangle.\,\mathtt{n}!\langle x, v\rangle\,) \;\sim_m\; \emptyset \tag{3}$$

where, for the sake of presentation, we exploit the context $\emptyset + [\![\cdot]\!] \triangleq [\mathtt{m}]\,(\mathtt{m}!\langle\rangle \mid \mathtt{m}?\langle\rangle + [\![\cdot]\!])$ and the term $\emptyset \triangleq [\mathtt{m}]\,(\mathtt{m}!\langle\rangle \mid \mathtt{m}?\langle\rangle)$. Communication along the private endpoint $\mathtt{m}$ models the τ action of π-calculus, while activities $\mathtt{n}?\langle x, v\rangle$ and $\mathtt{n}!\langle x, v\rangle$ recall the π-calculus actions $a(b)$ and $\bar{a}b$, respectively. Intuitively, the equality means that a service that emits the data it has received behaves as a service that simply performs an unobservable action. Although the two terms in (3) are syntactically different, it turns out that they are bisimilar. Indeed, the only transition that the term $\emptyset$ can perform is:

$$\emptyset \xrightarrow{\emptyset} \mathbf{0}$$

Trivially, the other term can reply by executing the activity on the left-hand side of +:

$$[x]\,(\,\emptyset + \mathtt{n}?\langle x, v\rangle.\,\mathtt{n}!\langle x, v\rangle\,) \xrightarrow{\emptyset} \mathbf{0}$$

Moreover, the term on the left in (3) can also perform the following transition:

$$[x]\,(\,\emptyset + \mathtt{n}?\langle x, v\rangle.\,\mathtt{n}!\langle x, v\rangle\,) \xrightarrow{\mathtt{n} \rhd [x]\,\langle x,v\rangle} [\mathtt{m}]\,(\mathtt{m}!\langle\rangle \mid \mathtt{n}!\langle x, v\rangle)$$

To this, $\emptyset$ can reply with an $\emptyset$-transition and, then, for all v', $[\mathtt{m}]\,(\mathtt{m}!\langle\rangle \mid \mathtt{n}!\langle v', v\rangle)$ and $(\mathbf{0} \mid \mathtt{n}!\langle v', v\rangle)$ are bisimilar. Indeed, both of them can only perform a transition labelled by $\mathtt{n} \lhd \langle v', v\rangle$ and evolve to $[\mathtt{m}]\,\mathtt{m}!\langle\rangle$ and $\mathbf{0}$, respectively, that cannot perform any further transition.

As another example regarding labelled bisimilarity, we can now prove that

$$[\mathtt{n}]\,(\,\mathtt{m}!\langle\mathtt{n}\rangle \mid \mathtt{n}!\langle\rangle\,) \sim_m [\mathtt{n}]\;\mathtt{m}!\langle\mathtt{n}\rangle$$

In fact, the family of relations $\{\mathcal{R}_\emptyset, \mathcal{R}_{\{\mathtt{n}\}}\}$, where $\mathcal{R}_\emptyset = \{\,(\,[\mathtt{n}]\,(\,\mathtt{m}!\langle\mathtt{n}\rangle \mid \mathtt{n}!\langle\rangle\,),\; [\mathtt{n}]\;\mathtt{m}!\langle\mathtt{n}\rangle\,)\,\}$ and $\mathcal{R}_{\{\mathtt{n}\}} = \{\,(\,\mathtt{n}!\langle\rangle\,,\; \mathbf{0}\,)\,\}$, is a labelled bisimulation.

We want now to prove that labelled bisimilarity is a congruence for μCOWS^m. To this aim we also need to consider *open* terms, i.e. terms with free variables, although we have only defined (strong) labelled bisimulation and bisimilarity over closed terms. We proceed as in [36, Section 4.4], where a similar situation is faced in the setting of CCS. Therefore, we extend the definition of $\sim_m^{\mathcal{N}}$ as follows.

Definition 5. *Let s_1 and s_2 be two μCOWS^m terms containing free variables $\bar{x}$ at most. Then $s_1 \sim_m^{\mathcal{N}} s_2$ if, for all values $\bar{v}$ such that $|\bar{x}|=|\bar{v}|$, $s_1 \cdot \{\bar{x} \mapsto \bar{v}\} \sim_m^{\mathcal{N}} s_2 \cdot \{\bar{x} \mapsto \bar{v}\}$.*

In other words, $\sim_m^{\mathcal{N}}$ is generalised to open terms as an hyperequivalence.

To prove the congruence result, we introduce also the notion of *bisimulation up-to structural congruence*: it is defined as a labelled bisimulation except for the fact that the $\mathcal{R}_{\mathcal{N}}$ in the three clauses of Definition 4 is replaced by the (compound) relation $\equiv \mathcal{R}_{\mathcal{N}} \equiv$. The next lemma shows that a bisimulation up-to $\equiv$ can be used as a sound proof-technique for labelled bisimulation. In the sequel, for the sake of simplicity, we explicitly write index $\mathcal{N}$ in a relation $\mathcal{R}_{\mathcal{N}}$ only when it is necessary or is modified in the considered case. Moreover, for the sake of readability, we only outline here the techniques used in the proofs and refer the interested reader to Appendix A for a full account.

Lemma 1. *Let $\mathcal{F}$ be a labelled bisimulation up-to $\equiv$; then, $\mathcal{F}$ is a labelled bisimulation.*

Proof. We need to prove that the family of relations

$$\{\, \{\, (s_1, s_2) \;:\; s_1 \equiv s_1',\ s_2' \equiv s_2,\ s_1' \mathcal{R} s_2' \,\} \;:\; \mathcal{R} \in \mathcal{F} \,\}$$

is a labelled bisimulation. The key point of the proof is that if $s_1 \xrightarrow{\alpha} s'$ then, by rule *(cong)* and since $s_1 \equiv s_1'$, we get that $s_1' \xrightarrow{\alpha} s'$. The proof then proceeds by case analysis on α, exploiting the fact that $s_1' \mathcal{R} s_2'$. □

Theorem 1. *$\sim_m$ is a congruence for μCOWS^m closed terms.*

Proof (sketch). We shall prove that, given two μCOWS^m closed terms s_1 and s_2, if $s_1 \sim_m s_2$ then $\mathbb{C}[\![s_1]\!] \sim_m \mathbb{C}[\![s_2]\!]$ for every (possibly open) context $\mathbb{C}$. The proof is by induction on the structure of the context $\mathbb{C}$. □

It is worth noticing that Theorem 1 does not hold in case of open terms. For example, consider the following equation:

$$\mathbf{0} + \mathtt{n}?\langle x\rangle.\, \mathtt{n}!\langle x\rangle \;\sim_m\; [\mathtt{m}]\,(\mathtt{m}!\langle x\rangle \mid \mathtt{m}?\langle x\rangle)$$

Although the two terms above are open, we can easily prove that they are bisimilar. Indeed, we can prove that $\mathbf{0} + \mathtt{n}?\langle v\rangle.\, \mathtt{n}!\langle v\rangle \;\sim_m\; [\mathtt{m}]\,(\mathtt{m}!\langle v\rangle \mid \mathtt{m}?\langle v\rangle)$ holds for all v (we can proceed as for (3)). However, the context $[x]\,[\![\cdot]\!]$ can tell the two terms apart, since $[x]\,(\mathbf{0} + \mathtt{n}?\langle x\rangle.\, \mathtt{n}!\langle x\rangle)$ can perform action $\mathtt{n} \rhd [x]\,\langle x\rangle$, while $[x, \mathtt{m}]\,(\mathtt{m}!\langle x\rangle \mid \mathtt{m}?\langle x\rangle)$ cannot perform any transition.

Now, we prove that open barbed bisimilarity and labelled bisimilarity coincide. In other words, labelled bisimilarity is *sound* and *complete* with respect to the barbed (contextual) one.

Theorem 2 (Soundness of $\sim_m$ w.r.t. $\simeq_m$). *Given two μCOWS^m closed terms s_1 and s_2, if $s_1 \sim_m s_2$ then $s_1 \simeq_m s_2$.*

Proof (sketch). By Theorem 1, we have that $\sim_m$ is context closed. Thus, we only need to prove that $\sim_m$ is barb preserving and computation closed. □

Theorem 3 (Completeness of $\sim_m$ w.r.t. $\simeq_m$). *Given two μCOWS^m closed terms s_1 and s_2, if $s_1 \simeq_m s_2$ then $s_1 \sim_m s_2$.*

Proof (sketch). We define a family of relations $\mathcal{F} = \{\mathcal{R}_\mathcal{N} \ : \ \mathcal{N} \text{ set of names}\}$ such that $\simeq_m$ is included in $\mathcal{R}_\emptyset$ and show that it is a labelled bisimulation. Let $\mathcal{N}$ be the set $\{n_1, \ldots, n_m\}$, then $s_1 \mathcal{R}_\mathcal{N} s_2$ if there exist $\mathtt{m}_1, \ldots, \mathtt{m}_m$ fresh such that

$$[n_1, \ldots, n_m]\,(s_1 \mid \mathtt{m}_1!\langle n_1\rangle \mid \ldots \mid \mathtt{m}_m!\langle n_m\rangle) \simeq_m [n_1, \ldots, n_m]\,(s_2 \mid \mathtt{m}_1!\langle n_1\rangle \mid \ldots \mid \mathtt{m}_m!\langle n_m\rangle)$$

Take $s_1 \mathcal{R}_\mathcal{N} s_2$ and a transition $s_1 \xrightarrow{\alpha} s_1'$; then, the proof proceeds by case analysis on α. □

Corollary 1. *$\sim_m$ and $\simeq_m$ coincide.*

Proof. Directly from Theorems 2 and 3. □

Weak open barbed and labelled bisimilarities. Our semantic theories extend in a standard way to the weak case. Therefore, weak transitions are defined as follows: $\Longrightarrow$ means $(\xrightarrow{\emptyset})^*$, i.e. zero or more $\emptyset$-transitions; $\overset{\alpha}{\Longrightarrow}$ means $\Longrightarrow\xrightarrow{\alpha}\Longrightarrow$; $\overset{\hat{\alpha}}{\Longrightarrow}$ means $\overset{\alpha}{\Longrightarrow}$ if $\alpha \neq \emptyset$ and $\Longrightarrow$ if $\alpha = \emptyset$. Predicate $s \Downarrow_\mathtt{n}$ holds true if there exist s' such that $s \Longrightarrow s'$ and $s' \downarrow_\mathtt{n}$.

Then, *weak open barbed bisimilarity*, written $\cong_m$, is obtained by replacing $s \downarrow_\mathtt{n}$ with $s \Downarrow_\mathtt{n}$, and $\xrightarrow{\emptyset}$ with $\Longrightarrow$ in Definition 2.

To define weak labelled bisimilarity we replace $\xrightarrow{\alpha}$ with $\overset{\hat{\alpha}}{\Longrightarrow}$ in the three clauses of Definition 4. As for π-calculus, the only subtle point is the case of receive actions, which have to be simulated by $\Longrightarrow\xrightarrow{\mathtt{n} \rhd [\bar{x}]\,\bar{w}}\Longrightarrow$ and requires the substitution to be applied immediately after the transition $\xrightarrow{\mathtt{n} \rhd [\bar{x}]\,\bar{w}}$, i.e. before further $\emptyset$-transitions (otherwise, just like in π-calculus, the resulting relation would not be an equivalence).

Definition 6 (Weak labelled bisimilarity). *A names-indexed family of relations $\{\mathcal{R}_\mathcal{N}\}_\mathcal{N}$ is a* weak labelled bisimulation *if, whenever $s_1 \mathcal{R}_\mathcal{N} s_2$ and $s_1 \xrightarrow{\alpha} s_1'$, where* $\mathrm{bu}(\alpha)$ *are fresh, then:*

1. *if $\alpha = \mathtt{n} \rhd [\bar{x}]\,\bar{w}$ then one of the following holds:*
   (a) $\exists\, s_2'' \ : \ s_2 \Longrightarrow\xrightarrow{\mathtt{n} \rhd [\bar{x}]\,\bar{w}} s_2''$ *and*
   $\forall\, \bar{v}$ *s.t.* $\mathcal{M}(\bar{x}, \bar{v}) = \sigma\ \exists\, s_2' \ : \ s_2'' \cdot \sigma \Longrightarrow s_2'$ *and* $s_1' \cdot \sigma \mathcal{R}_\mathcal{N}\, s_2'$
   (b) $\exists\, s_2' \ : \ s_2 \Longrightarrow s_2'$ *and* $\forall\, \bar{v}$ *s.t.* $\mathcal{M}(\bar{x}, \bar{v}) = \sigma \ : \ s_1' \cdot \sigma \mathcal{R}_\mathcal{N}\, (s_2' \mid \mathtt{n}!(\bar{w} \cdot \sigma))$
2. *if $\alpha = \mathtt{n} \lhd [\bar{n}]\,\bar{v}$ where $\mathtt{n} \notin \mathcal{N}$ then $\exists\, s_2' \ : \ s_2 \overset{\mathtt{n} \lhd [\bar{n}]\,\bar{v}}{\Longrightarrow} s_2'$ and $s_1' \mathcal{R}_{\mathcal{N} \cup \bar{n}}\, s_2'$*
3. *if $\alpha = \emptyset$ then $\exists\, s_2' \ : \ s_2 \Longrightarrow s_2'$ and $s_1' \mathcal{R}_\mathcal{N}\, s_2'$*

Two closed terms s_1 and s_2 are weak $\mathcal{N}$-bisimilar*, written $s_1 \approx_m^{\mathcal{N}} s_2$, if $s_1 \mathcal{R}_{\mathcal{N}} s_2$ for some $\mathcal{R}_{\mathcal{N}}$ in a weak labelled bisimulation. They are* weak labelled bisimilar*, written $s_1 \approx_m s_2$, if they are weak $\emptyset$-bisimilar. $\approx_m^{\mathcal{N}}$ is called weak $\mathcal{N}$-bisimilarity, while $\approx_m$ is called weak labelled bisimilarity.*

Results of congruence and coincidence still hold for the weak case. We omit the corresponding proofs because they do not require new techniques and, indeed, are the standard generalisation of the strong ones.

We conclude with an example inspired to the law $!(a(b).\,\bar{a}b) = \mathbf{0}$ that holds for weak bisimilarity in asynchronous π-calculus [3]. In fact, the analogous of equality (3) for the weak case is:

$$*\,[x]\,\mathtt{n}?\langle x, v\rangle.\,\mathtt{n}!\langle x, v\rangle \approx_m \mathbf{0} \tag{4}$$

To prove validity, the most significant case is simulating the transition

$$*\,[x]\,\mathtt{n}?\langle x, v\rangle.\,\mathtt{n}!\langle x, v\rangle \xrightarrow{\mathtt{n} \rhd [x]\,\langle x,v\rangle} *\,[x]\,(\mathtt{n}?\langle x, v\rangle.\,\mathtt{n}!\langle x, v\rangle) \mid \mathtt{n}!\langle x, v\rangle$$

The term on the right of (4) replies with an empty transition (i.e. it does not perform any action) and it is easy to show that, for all v', $(*\,[x]\,(\mathtt{n}?\langle x, v\rangle.\,\mathtt{n}!\langle x, v\rangle) \mid \mathtt{n}!\langle v', v\rangle)$ and $(\mathbf{0} \mid \mathtt{n}!\langle v', v\rangle)$ are weak bisimilar.

4 μCOWS : the protection- and kill-free fragment of COWS

The fragment of COWS presented in this section, namely μCOWS, dispenses with those activities dealing with termination, i.e. kill and protection. In other words, μCOWS extends μCOWS^m with priority in the parallel composition.

4.1 Syntax and operational semantics

The syntax of μCOWS and the set of laws defining the structural congruence coincide with that of μCOWS^m, shown in Tables 1 and 2, respectively. Instead, the labelled transition relation $\xrightarrow{\alpha}$ is the least relation over μCOWS services induced by the rules in Tables 4 and 5, where rules *(del$_2$)*, *(com$_2$)* and *(par$_2$)* replace *(del)*, *(com)* and *(par)*, respectively. Labels are now generated by the following grammar:

$$\alpha \;::=\; \mathtt{n} \lhd [\bar{n}]\,\bar{v} \;\mid\; \mathtt{n} \rhd [\bar{x}]\,\bar{w} \;\mid\; \sigma \;\mid\; \mathtt{n}\,\sigma\,\ell\,\bar{v}$$

The new label $\mathtt{n}\,\sigma\,\ell\,\bar{v}$ denotes execution of a communication over $\mathtt{n}$ with matching values $\bar{v}$, generated substitution having ℓ pairs, and substitution σ to be still applied. We use $\mathrm{ce}(\alpha)$ to denote the names composing the endpoint in case α denotes execution of a communication, i.e. $\mathrm{ce}(\alpha)$ is $\emptyset$ except for $\alpha = \mathtt{n}\,\sigma\,\ell\,\bar{v}$ for which we let $\mathrm{ce}(\mathtt{n}\,\sigma\,\ell\,\bar{v}) = \mathtt{n}$. Moreover, for $\alpha = \mathtt{n}\,\sigma\,\ell\,\bar{v}$ we let $\mathrm{u}(\mathtt{n}\,\sigma\,\ell\,\bar{v}) = \mathrm{u}(\sigma)$. Now, *computational steps* are denoted both by label of the form $\mathtt{n}\,\emptyset\,\ell\,\bar{v}$ and $\emptyset$.

The definition of the labelled transition relation exploits a new auxiliary predicate $\mathrm{noConf}(s, \mathtt{n}, \bar{v}, \ell)$, with ℓ natural number. The predicate, defined in Table 6, holds true if s cannot immediately perform a receive over the endpoint $\mathtt{n}$ matching $\bar{v}$ and generating a substitution σ with $|\sigma| < \ell$.

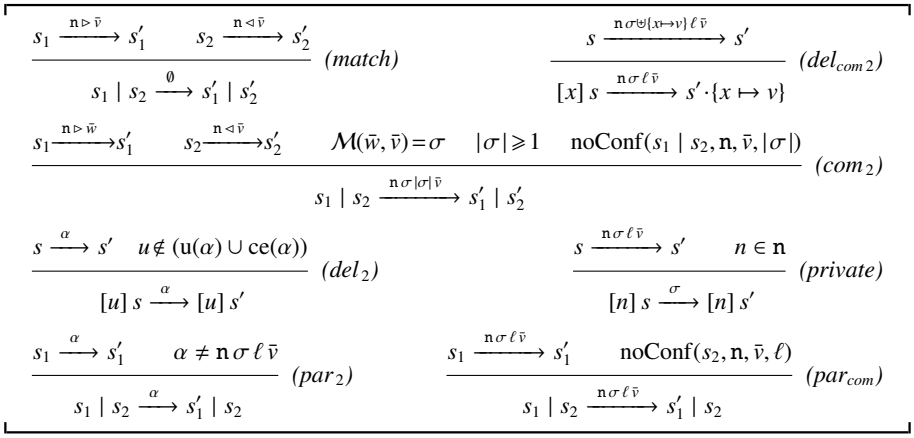

Table 5. μCOWS operational semantics (additional rules)

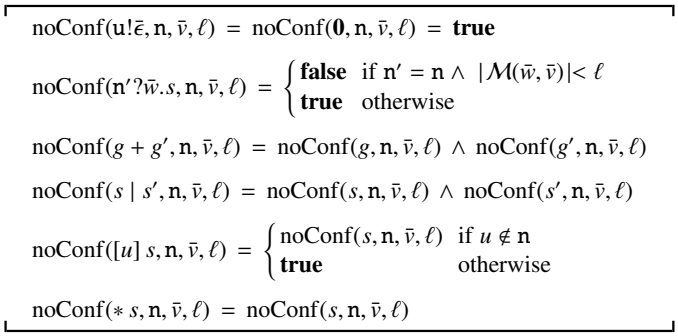

Table 6. There are not conflicting receives along $\mathtt{n}$ matching $\bar{v}$

We only comment on salient points. In μCOWS, two different kinds of communication labels can be generated: σ and $\mathtt{n}\,\sigma\,\ell\,\bar{v}$. The latter label, produced by rule *(com$_2$)*, carries information about the communication that has taken place (i.e. the endpoint, the transmitted values, the generated substitution and its length) used to check the presence of conflicting receives in parallel components. Indeed, if more than one matching is possible, the receive that needs fewer substitutions is selected to progress (rules *(com$_2$)* and *(par$_{com}$)*). This mechanism permits to correlate different service communications thus implicitly creating interaction sessions and can be exploited to model the precedence of a service instance over the corresponding service specification when both can process the same request. The check for presence of a conflict is not needed when either the performed receive has the highest priority (i.e. the substitution has length 0) or the communication takes place along a private endpoint. In the former case, label $\emptyset$ is immediately generated by rule *(match)*. In the latter case, when the delimitation of a name belonging to the endpoint of a communication label is encountered (i.e. the communication is identified as private), the transition label $\mathtt{n}\,\sigma\,\ell\,\bar{v}$ is turned into σ (rule *(private)*).

Rule *(del$_{com\,2}$)* is similar to *(del$_{com}$)* (shown in Table 4) but deals with labels generated by communications still subject to priority check. Notably, during the inference of a transition labelled by $\mathtt{n}\,\sigma\,\ell\,\bar{v}$, the length of the substitution to be applied decreases (rule *(del$_{com\,2}$)*), while the length ℓ of the initial substitution does never change, which makes it suitable to check, in any moment, existence of better matching, i.e. of parallel receives with greater priority. Execution of parallel services is interleaved (rule *(par$_2$)*), but when a communication is performed. In such case, the progress of the receive activity with greater priority must be ensured.

4.2 Observational semantics

As we have done for μCOWS^m in the previous section, we define now open barbed and labelled bisimilarities for μCOWS.

Strong open barbed bisimilarity. The notion of basic observable defined for μCOWS^m (Definition 1) can be used also to define the open barbed bisimilarity for μCOWS.

Definition 7 (Open barbed bisimilarity). *A symmetric binary relation $\mathcal{R}$ on* μCOWS *closed terms is an* open barbed bisimulation *if whenever $s_1 \mathcal{R} s_2$ the following holds:*

(Barb preservation) *if $s_1 \downarrow_{\mathtt{n}}$ then $s_2 \downarrow_{\mathtt{n}}$;*

(Computation closure) *if $s_1 \xrightarrow{\emptyset} s_1'$ (resp. $s_1 \xrightarrow{\mathtt{n}\emptyset\ell\bar{v}} s_1'$) then there exists s_2' such that $s_2 \xrightarrow{\emptyset} s_2'$ (resp. $s_2 \xrightarrow{\mathtt{n}\emptyset\ell\bar{v}} s_2'$ or $\ell = |\bar{v}| \wedge s_2 \xrightarrow{\emptyset} s_2'$) and $s_1' \mathcal{R} s_2'$;*

(Context closure) *$\mathbb{C}[\![s_1]\!]\,\mathcal{R}\,\mathbb{C}[\![s_2]\!]$, for every closed context $\mathbb{C}$.*

Two closed terms s_1 and s_2 are open barbed bisimilar*, written $s_1 \simeq_\mu s_2$, if $s_1 \mathcal{R} s_2$ for some open barbed bisimulation $\mathcal{R}$. $\simeq_\mu$ is called open barbed bisimilarity.*

The major difference with the definition of barbed bisimilarity for μCOWS^m (Definition 2) is that here we also take care of computations of the form $\mathtt{n}\,\emptyset\,\ell\,\bar{v}$.

We show now that in μCOWS it is not true any longer that receive activities are always unobservable. In fact, in Section 3 we have shown that, for μCOWS^m, $\sim_m$ enjoys the equality (3). Hence, by Theorem 3, we get that $[x]\,(\,\emptyset + \mathtt{n}?\langle x, v\rangle.\,\mathtt{n}!\langle x, v\rangle\,) \simeq_m \emptyset$, which means that receive activities cannot be observed (similarly to asynchronous π-calculus). In μCOWS, however, the context $\mathbb{C} \triangleq [y, z]\,\mathtt{n}?\langle y, z\rangle.\,\mathtt{m}!\langle\rangle \mid \mathtt{n}!\langle v', v\rangle \mid [\![\cdot]\!]$ can tell the two terms above apart. In fact, we have

$$\mathbb{C}[\![\emptyset]\!] \xrightarrow{\mathtt{n}\emptyset 2\langle v', v\rangle} \mathtt{m}!\langle\rangle \mid \emptyset$$

where the term $(\mathtt{m}!\langle\rangle \mid \emptyset)$ satisfies the predicate $\downarrow_{\mathtt{m}}$. Instead, the other term cannot properly reply because the receive $\mathtt{n}?\langle x, v\rangle$ has higher priority than $\mathtt{n}?\langle y, z\rangle$ when synchronising with the invocation $\mathtt{n}!\langle v', v\rangle$. Thus, $\mathbb{C}[\![[x]\,(\,\emptyset + \mathtt{n}?\langle x, v\rangle.\,\mathtt{n}!\langle x, v\rangle\,)]\!]$ can only evolve to terms that cannot immediately satisfy the predicate $\downarrow_{\mathtt{m}}$. From this, we have

$$[x]\,(\,\emptyset + \mathtt{n}?\langle x, v\rangle.\,\mathtt{n}!\langle x, v\rangle\,) \not\simeq_\mu \emptyset \tag{5}$$

This means that, receive activities that exercise a priority (i.e. receives whose arguments contain some values) can be detected by an interacting observer.

Now, consider the term $[x, x']\,(\,\emptyset + \mathtt{n}?\langle x, x'\rangle.\,\mathtt{n}!\langle x, x'\rangle\,)$. Since $\mathtt{n}?\langle x, x'\rangle$ does not exercise any priority on parallel terms, we have that

$$[x, x']\,(\,\emptyset + \mathtt{n}?\langle x, x'\rangle.\,\mathtt{n}!\langle x, x'\rangle\,) \simeq_\mu \emptyset \qquad \mathbb{C}[\![[x, x']\,(\,\emptyset + \mathtt{n}?\langle x, x'\rangle.\,\mathtt{n}!\langle x, x'\rangle\,)]\!] \simeq_\mu \mathbb{C}[\![\emptyset]\!]$$

For similar reasons, we have that $\emptyset + \mathtt{n}?\langle\rangle.\,\mathtt{n}!\langle\rangle \simeq_\mu \emptyset$ and $\mathbb{D}[\![\emptyset + \mathtt{n}?\langle\rangle.\,\mathtt{n}!\langle\rangle]\!] \simeq_\mu \mathbb{D}[\![\emptyset]\!]$ for $\mathbb{D} \triangleq \mathtt{n}?\langle\rangle.\,\mathtt{m}!\langle\rangle \mid \mathtt{n}!\langle\rangle \mid [\![\cdot]\!]$.

Therefore, differently from μCOWS^m, communication in μCOWS is neither purely asynchronous nor purely synchronous. Indeed, receives having the smallest priority (i.e. whose arguments are, possible empty, tuples of variables) cannot be observed, while, by exploiting proper contexts, the other receives can be detected.

Strong labelled bisimilarity. For what we have seen in the previous paragraph, the equivalence $\sim_m$ is not suitable for characterising the open barbed bisimilarity for μCOWS. Indeed, $\sim_m$ enjoys equality (3) while $\simeq_\mu$ enjoys disequality (5). As consequence, $\sim_m$ is not preserved by all closed language contexts (which would prevent compositional reasoning).

Thus, we provide a new notion of bisimulation defined on top of the labelled transition system defining the semantics of μCOWS.

Definition 8 (Labelled bisimilarity). *A names-indexed family of relations $\{\mathcal{R}_\mathcal{N}\}_\mathcal{N}$ is a* labelled bisimulation *if, whenever $s_1 \mathcal{R}_\mathcal{N} s_2$ and $s_1 \xrightarrow{\alpha} s_1'$, where* $\mathrm{bu}(\alpha)$ *are fresh, then:*

1. *if $\alpha = \mathtt{n} \rhd [\bar{x}]\,\bar{w}$ then one of the following holds:*
   (a) $\exists\, s_2' \,:\, s_2 \xrightarrow{\mathtt{n}\rhd[\bar{x}]\,\bar{w}} s_2'$ *and*
   $\forall\, \bar{v}$ *s.t.* $\mathcal{M}(\bar{x}, \bar{v}) = \sigma$ *and* $\mathrm{noConf}(s_2, \mathtt{n}, \bar{w}\cdot\sigma, |\bar{x}|) \,:\, s_1'\cdot\sigma\, \mathcal{R}_\mathcal{N}\, s_2'\cdot\sigma$
   (b) $|\bar{x}|{=}|\bar{w}|$ *and* $\exists\, s_2' : s_2 \xrightarrow{\emptyset} s_2'$ *and*
   $\forall\, \bar{v}$ *s.t.* $\mathcal{M}(\bar{x}, \bar{v}) = \sigma$ *and* $\mathrm{noConf}(s_2, \mathtt{n}, \bar{w}\cdot\sigma, |\bar{x}|) \,:\, s_1'\cdot\sigma\, \mathcal{R}_\mathcal{N}\, (s_2' \mid \mathtt{n}!\bar{v})$
2. *if $\alpha = \mathtt{n}\,\emptyset\,\ell\,\bar{v}$ where $\ell = |\bar{v}|$ then one of the following holds:*
   (a) $\exists\, s_2' \,:\, s_2 \xrightarrow{\mathtt{n}\,\emptyset\,\ell\,\bar{v}} s_2'$ *and* $s_1'\, \mathcal{R}_\mathcal{N}\, s_2'$ (b) $\exists\, s_2' \,:\, s_2 \xrightarrow{\emptyset} s_2'$ *and* $s_1'\, \mathcal{R}_\mathcal{N}\, s_2'$
3. *if $\alpha = \mathtt{n} \lhd [\bar{n}]\,\bar{v}$ where $\mathtt{n} \notin \mathcal{N}$ then $\exists\, s_2' \,:\, s_2 \xrightarrow{\mathtt{n}\lhd[\bar{n}]\,\bar{v}} s_2'$ and $s_1'\, \mathcal{R}_{\mathcal{N}\cup\bar{n}}\, s_2'$*
4. *if $\alpha = \emptyset$ or $\alpha = \mathtt{n}\,\emptyset\,\ell\,\bar{v}$, where $\ell \neq |\bar{v}|$, then $\exists\, s_2' \,:\, s_2 \xrightarrow{\alpha} s_2'$ and $s_1'\, \mathcal{R}_\mathcal{N}\, s_2'$*

Two closed terms s_1 and s_2 are $\mathcal{N}$-bisimilar*, written $s_1 \sim_\mu^\mathcal{N} s_2$, if $s_1 \mathcal{R}_\mathcal{N} s_2$ for some $\mathcal{R}_\mathcal{N}$ in a labelled bisimulation. They are* labelled bisimilar*, written $s_1 \sim_\mu s_2$, if they are $\emptyset$-bisimilar. $\sim_\mu^\mathcal{N}$ is called $\mathcal{N}$-bisimilarity, while $\sim_\mu$ is called labelled bisimilarity.*

Clause 1 deals with both observable and unobservable receives. In fact, all receives can be simulated in a normal way (clause 1.(a)); additionally, receives such that $|\bar{x}|{=}|\bar{w}|$, i.e. $\bar{w}$ contains only variables or is the empty tuple (since $\bar{x} \subseteq \bar{w}$ and $\bar{w}\backslash\bar{x}$ does not contain variables), can be simulated by an internal action leading to a term that, when composed with the invoke activity consumed by the receive, stands in the appropriate relation (clause 1.(b)). For similar reasons, clause 2 permits replying with an $\emptyset$-transition

to communications involving an unobservable receive ($\ell =| \bar{v} |$ implies that the tuple argument of the receive is either empty or only contains variables). Indeed, such clause is explained by the following equation: $\mathbb{C}[\![[x, x'](\emptyset + \mathtt{n}?\langle x, x'\rangle.\, \mathtt{n}!\langle x, x'\rangle)]\!] \sim_\mu \mathbb{C}[\![\emptyset]\!]$ for $\mathbb{C} \triangleq [\![\cdot]\!] \mid \mathtt{n}!\langle v', v\rangle$. In fact, if we do not have that clause, we would have that

$$\mathbb{C}[\![[x, x'](\emptyset + \mathtt{n}?\langle x, x'\rangle.\, \mathtt{n}!\langle x, x'\rangle)]\!] \xrightarrow{\mathtt{n}\,\emptyset\,2\,\langle v', v\rangle} \quad \text{and} \quad \mathbb{C}[\![\emptyset]\!] \xrightarrow{\mathtt{n}\,\emptyset\,2\,\langle v', v\rangle} \not\longrightarrow$$

which would imply that $\mathbb{C}[\![[x, x'](\emptyset + \mathtt{n}?\langle x, x'\rangle.\, \mathtt{n}!\langle x, x'\rangle)]\!] \not\sim_\mu \mathbb{C}[\![\emptyset]\!]$.

Differently from labelled bisimulation for μCOWS^m, here in clause 1 the two continuations are not related for any tuple of values, but only for those tuples that can be effectively received (i.e. that do not give rise to communication conflicts). Indeed, for example, the following two μCOWS terms:

$$s_1 \triangleq [x](\mathtt{n}?\langle v\rangle \;+\; \mathtt{n}?\langle x\rangle.\, [\mathtt{m}](\mathtt{m}!\langle x\rangle \mid \mathtt{m}?\langle x\rangle + \mathtt{m}?\langle v\rangle.\, \mathtt{m}'!\langle\rangle))$$
$$s_2 \triangleq [x](\mathtt{n}?\langle v\rangle \;+\; \mathtt{n}?\langle x\rangle.\, [\mathtt{m}](\mathtt{m}!\langle x\rangle \mid \mathtt{m}?\langle x\rangle))$$

are both barbed and labelled bisimilar. Instead, if the definition of bisimulation does not have condition noConf$(s_2, \mathtt{n}, \bar{w}\cdot\sigma, |\,\bar{x}\,|)$ in clause 1, we would have that $s_1 \simeq_\mu s_2$ but $s_1 \not\sim_\mu s_2$. Indeed, after execution of action $\mathtt{n} \triangleright [\bar{x}]\,\bar{x}$, it is not correct to consider the continuations under substitutions $\{x \mapsto v'\}$ for all v', because such action cannot be performed in case the received value v' is v (it is pre-empted by $\mathtt{n}?\langle v\rangle$).

In clauses 1.(b) and 2.(b) we require that s_2 can only reply with an $\emptyset$-transition and not with a transition labelled by $\mathtt{m}\,\emptyset\,\ell\,\bar{v}$. Otherwise, the relation would not be preserved by parallel composition, because communication $\mathtt{m}\,\emptyset\,\ell\,\bar{v}$ is subject to conflict check and, hence, could be blocked by a receive with higher priority performed by a parallel term.

Some illustrative equalities follow.

1. $\mathtt{n}!\langle\rangle \mid \mathtt{n}?\langle\rangle \;\not\sim_\mu\; \mathtt{m}!\langle\rangle \mid \mathtt{m}?\langle\rangle$
2. $[\mathtt{n}](\mathtt{n}!\langle\rangle \mid \mathtt{n}?\langle\rangle) \;\sim_\mu\; [\mathtt{m}](\mathtt{m}!\langle\rangle \mid \mathtt{m}?\langle\rangle) \;\sim_\mu\; [\mathtt{m}](\mathtt{m}!\langle\bar{v}\rangle \mid [\bar{x}]\,\mathtt{m}?\langle\bar{x}\rangle)$
3. $[x](\emptyset + \mathtt{n}?\langle x, v\rangle.\, \mathtt{n}!\langle x, v\rangle) \;\not\sim_\mu\; \emptyset$
4. $[x, x'](\emptyset + \mathtt{n}?\langle x, x'\rangle.\, \mathtt{n}!\langle x, x'\rangle) \;\sim_\mu\; \emptyset$
5. $\emptyset + \mathtt{n}?\langle\rangle.\, \mathtt{n}!\langle\rangle \;\sim_\mu\; \emptyset$
6. $\mathbb{C}[\![[x, x'](\emptyset + \mathtt{n}?\langle x, x'\rangle.\, \mathtt{n}!\langle x, x'\rangle)]\!] \sim_\mu \mathbb{C}[\![\emptyset]\!]$ for $\mathbb{C} \triangleq [y, z]\,\mathtt{n}?\langle y, z\rangle.\, \mathtt{n}'!\langle\rangle \mid \mathtt{n}!\langle v', v\rangle \mid [\![\cdot]\!]$

Remark 1 (On computational steps). μCOWS supports two different kinds of computational steps, labelled by $\emptyset$ and $\mathtt{n}\,\emptyset\,\ell\,\bar{v}$. To be a congruence, labelled bisimilarity deals with them separately. Indeed, let τ denote any computational step, and suppose to weaken Definition 8 by replacing $\emptyset$ with τ in clause 1, by removing clause 2, and by modifying clause 4 in order to only consider the case $\alpha = \tau$. We would have that $*\,\mathtt{n}!\langle v\rangle \mid \emptyset \;\sim_\mu\; *\,\mathtt{n}!\langle v\rangle \mid [x]\,\mathtt{n}?\langle x\rangle.\, \mathtt{n}!\langle x\rangle$. In fact, the relevant cases are:

- if $*\,\mathtt{n}!\langle v\rangle \mid \emptyset \xrightarrow{\emptyset} *\,\mathtt{n}!\langle v\rangle$ then $*\,\mathtt{n}!\langle v\rangle \mid [x]\,\mathtt{n}?\langle x\rangle.\, \mathtt{n}!\langle x\rangle \xrightarrow{\mathtt{n}\,\emptyset\,1\,\langle v\rangle} *\,\mathtt{n}!\langle v\rangle \mid \mathtt{n}!\langle v\rangle \equiv *\,\mathtt{n}!\langle v\rangle$, and vice versa;
- if $*\,\mathtt{n}!\langle v\rangle \mid [x]\,\mathtt{n}?\langle x\rangle.\, \mathtt{n}!\langle x\rangle \xrightarrow{\mathtt{n}\triangleright[\langle x\rangle]\,\langle x\rangle} *\,\mathtt{n}!\langle v\rangle \mid \mathtt{n}!\langle x\rangle$ then $*\,\mathtt{n}!\langle v\rangle \mid \emptyset \xrightarrow{\emptyset} *\,\mathtt{n}!\langle v\rangle$ and for all v' holds that $*\,\mathtt{n}!\langle v\rangle \mid \mathtt{n}!\langle x\rangle \cdot \{x \mapsto v'\}$ and $*\,\mathtt{n}!\langle v\rangle \mid \mathtt{n}!\langle v'\rangle$ are bisimilar.

However, the context $\mathtt{n}?\langle v\rangle \mid [\![\cdot]\!]$ can tell the two terms apart, because the transition labelled $\mathtt{n}\,\emptyset\,1\,\langle v\rangle$ can be blocked by the activity $\mathtt{n}?\langle v\rangle$ (which has higher priority than $\mathtt{n}?\langle x\rangle$). This means that the modified labelled bisimilarity is not a congruence for μCOWS. Of course, the two terms above are not bisimilar according to Definition 8.

Now, we prove that labelled bisimilarity $\sim_\mu$ is a congruence for μCOWS. To this aim, besides the generalisation of $\sim_\mu$ to open terms and a lemma showing that a bisimulation up-to $\equiv$ can be used as a sound proof-technique for labelled bisimulation, we need a lemma establishing preservation of predicate noConf(_, _, _, _) by the relations belonging to a bisimulation. As usual, we only outline here the techniques used in the proofs and refer the interested reader to Appendix A for a full account.

Definition 9. *Let s_1 and s_2 be two μ*COWS *terms containing free variables $\bar{x}$ at most. Then $s_1 \sim_\mu^{\mathcal{N}} s_2$ if, for all values $\bar{v}$ such that $|\bar{x}|=|\bar{v}|$, $s_1\cdot\{\bar{x}\mapsto\bar{v}\} \sim_\mu^{\mathcal{N}} s_2\cdot\{\bar{x}\mapsto\bar{v}\}$.*

Lemma 2. *Let $\mathcal{F}$ be a labelled bisimulation up-to $\equiv$; then, $\mathcal{F}$ is a labelled bisimulation.*

Proof. The proof proceeds as the proof of Lemma 1. □

Lemma 3. *Let s_1 and s_2 be two μ*COWS *closed terms and $\mathcal{R}$ be a relation belonging to a labelled bisimulation such that $s_1\mathcal{R}s_2$. Then,* $\mathrm{noConf}(s_1,\mathtt{n},\bar{v},\ell) = \mathrm{noConf}(s_2,\mathtt{n},\bar{v},\ell)$ *for any $\mathtt{n}$, $\bar{v}$ and ℓ, with $\ell \leqslant |\bar{v}|$.*

Proof (sketch). The lemma is proved by contradiction. □

Theorem 4. *$\sim_\mu$ is a congruence for μ*COWS *closed terms.*

Proof (sketch). We shall prove that, given two μCOWS closed terms s_1 and s_2, if $s_1 \sim_\mu s_2$ then $\mathbb{C}[\![s_1]\!] \sim_\mu \mathbb{C}[\![s_2]\!]$ for every (possibly open) context $\mathbb{C}$. The proof is by induction on the structure of the context $\mathbb{C}$ and proceeds similarly to that of Theorem 1. □

Now, we prove that labelled bisimilarity is *sound* and *complete* with respect to open barbed bisimilarity.

Theorem 5 (Soundness of $\sim_\mu$ w.r.t. $\simeq_\mu$). *Given two μ*COWS *closed terms s_1 and s_2, if $s_1 \sim_\mu s_2$ then $s_1 \simeq_\mu s_2$.*

Proof (sketch). By Theorem 4, we have that $\sim_\mu$ is context closed. Thus, we only need to prove that $\sim_\mu$ is barb preserving and computation closed. □

Theorem 6 (Completeness of $\sim_\mu$ w.r.t. $\simeq_\mu$). *Given two μ*COWS *closed terms s_1 and s_2, if $s_1 \simeq_\mu s_2$ then $s_1 \sim_\mu s_2$.*

Proof (sketch). We define a family of relations $\mathcal{F} = \{\mathcal{R}_{\mathcal{N}} : \mathcal{N} \text{ set of names}\}$ such that $\simeq_\mu$ is included in $\mathcal{R}_\emptyset$, and show that it is a labelled bisimulation. Let $\mathcal{N}$ be the set $\{n_1,\ldots,n_m\}$, then $s_1\mathcal{R}_{\mathcal{N}}s_2$ if there exist $\mathtt{m}_1,\ldots,\mathtt{m}_m$ fresh such that

$$[n_1,\ldots,n_m]\,(s_1 \mid \mathtt{m}_1!\langle n_1\rangle \mid \ldots \mid \mathtt{m}_m!\langle n_m\rangle) \simeq_\mu [n_1,\ldots,n_m]\,(s_2 \mid \mathtt{m}_1!\langle n_1\rangle \mid \ldots \mid \mathtt{m}_m!\langle n_m\rangle)$$

Take $s_1\mathcal{R}_{\mathcal{N}}s_2$ and a transition $s_1 \xrightarrow{\alpha} s_1'$; then, the proof proceeds by case analysis on α. □

Corollary 2. *$\sim_\mu$ and $\simeq_\mu$ coincide.*

Proof. Directly from Theorems 5 and 6. □

Weak open barbed and labelled bisimilarities. As in Section 3.3, the observational semantic theories introduced for μCOWS extend in a standard way to the weak case.

Weak open barbed bisimilarity, namely $\cong_\mu$, is obtained by replacing $s \downarrow_{\mathrm{n}}$ with $s \Downarrow_{\mathrm{n}}$, $\xrightarrow{\emptyset}$ with $\Longrightarrow$, and $\xrightarrow{\mathrm{n}\emptyset\ell\bar{v}}$ with $\stackrel{\mathrm{n}\emptyset\ell\bar{v}}{\Longrightarrow}$ in Definition 7.

To define weak labelled bisimilarity we replace $\xrightarrow{\alpha}$ with $\stackrel{\hat{\alpha}}{\Longrightarrow}$ in the four clauses of Definition 8.

Definition 10 (Weak labelled bisimilarity). *A names-indexed family of relations* $\{\mathcal{R}_\mathcal{N}\}_\mathcal{N}$ *is a* weak labelled bisimulation *if, whenever* $s_1 \mathcal{R}_\mathcal{N} s_2$ *and* $s_1 \xrightarrow{\alpha} s_1'$, *where* $\mathrm{bu}(\alpha)$ *are fresh, then:*

1. *if* $\alpha = \mathrm{n} \triangleright [\bar{x}]\,\bar{w}$ *then one of the following holds:*
   (a) $\exists\, s_2'' \;:\; s_2 \stackrel{\mathrm{n}\triangleright[\bar{x}]\,\bar{w}}{\Longrightarrow} s_2''$ *and* $\forall\, \bar{v}$ *s.t.* $\mathcal{M}(\bar{x},\bar{v}) = \sigma$ *and* $\mathrm{noConf}(s_2, \mathrm{n}, \bar{w}\cdot\sigma, |\bar{x}|)$ $\exists\, s_2' \;:\; s_2''\cdot\sigma \Longrightarrow s_2'$ *and* $s_1'\cdot\sigma\, \mathcal{R}_\mathcal{N}\, s_2'$
   (b) $|\bar{x}| = |\bar{w}|$ *and* $\exists\, s_2' \;:\; s_2 \Longrightarrow s_2'$ *and* $\forall\, \bar{v}$ *s.t.* $\mathcal{M}(\bar{x},\bar{v}) = \sigma$ *and* $\mathrm{noConf}(s_2, \mathrm{n}, \bar{w}\cdot\sigma, |\bar{x}|) \;:\; s_1'\cdot\sigma\, \mathcal{R}_\mathcal{N}\, (s_2' \mid \mathrm{n}!\bar{v})$
2. *if* $\alpha = \mathrm{n}\emptyset\ell\bar{v}$ *where* $\ell = |\bar{v}|$ *then one of the following holds:*
   (a) $\exists\, s_2' \;:\; s_2 \stackrel{\mathrm{n}\emptyset\ell\bar{v}}{\Longrightarrow} s_2'$ *and* $s_1' \,\mathcal{R}_\mathcal{N}\, s_2'$ (b) $\exists\, s_2' \;:\; s_2 \Longrightarrow s_2'$ *and* $s_1' \,\mathcal{R}_\mathcal{N}\, s_2'$
3. *if* $\alpha = \mathrm{n} \triangleleft [\bar{n}]\,\bar{v}$ *where* $\mathrm{n} \notin \mathcal{N}$ *then* $\exists\, s_2' \;:\; s_2 \stackrel{\mathrm{n}\triangleleft[\bar{n}]\,\bar{v}}{\Longrightarrow} s_2'$ *and* $s_1' \,\mathcal{R}_{\mathcal{N}\cup\bar{n}}\, s_2'$
4. *if* $\alpha = \emptyset$ *or* $\alpha = \mathrm{n}\emptyset\ell\bar{v}$, *where* $\ell \neq |\bar{v}|$, *then* $\exists\, s_2' \;:\; s_2 \stackrel{\hat{\alpha}}{\Longrightarrow} s_2'$ *and* $s_1' \,\mathcal{R}\, s_2'$

Two closed terms s_1 *and* s_2 *are* weak $\mathcal{N}$-bisimilar, *written* $s_1 \approx_\mu^\mathcal{N} s_2$, *if* $s_1 \mathcal{R}_\mathcal{N} s_2$ *for some* $\mathcal{R}_\mathcal{N}$ *in a weak labelled bisimulation. They are* weak labelled bisimilar, *written* $s_1 \approx_\mu s_2$, *if they are weak* $\emptyset$*-bisimilar.* $\approx_\mu^\mathcal{N}$ *is called weak* $\mathcal{N}$*-bisimilarity, while* $\approx_\mu$ *is called weak labelled bisimilarity.*

The results of congruence and coincidence still hold for the weak case. We omit the corresponding proofs because they do not require new techniques and, indeed, are the standard generalisation of the strong ones.

We conclude with an example that is the analogous of equality (4) for μCOWS:

$$*\,[x,x']\,\mathrm{n}?\langle x,x'\rangle.\,\mathrm{n}!\langle x,x'\rangle \approx_\mu \mathbf{0}$$

To prove validity, the most significant case is simulating the transition

$$*\,[x,x']\,\mathrm{n}?\langle x,x'\rangle.\,\mathrm{n}!\langle x,x'\rangle \xrightarrow{\mathrm{n}\triangleright[x,x']\,\langle x,x'\rangle} *\,[x,x']\,(\mathrm{n}?\langle x,x'\rangle.\,\mathrm{n}!\langle x,x'\rangle) \mid \mathrm{n}!\langle x,x'\rangle$$

The term on the right replies with an empty transition and it is easy to show that, for all v and v', $(*\,[x,x']\,(\mathrm{n}?\langle x,x'\rangle.\,\mathrm{n}!\langle x,x'\rangle) \mid \mathrm{n}!\langle v,v'\rangle)$ and $(\mathbf{0} \mid \mathrm{n}!\langle v,v'\rangle)$ are weak bisimilar.

5 COWS

COWS is obtained by enriching μCOWS with two primitive operators to enable fault and compensation handling and guarantee transactional properties of services.

s ::=		(services)	g ::=		(receive-guarded choice)
	$\mathbf{kill}(k)$	(kill)		$\mathbf{0}$	(nil)
|	$u \bullet u'!\bar{\epsilon}$	(invoke)	|	$p \bullet o?\bar{w}.s$	(request processing)
|	g	(receive-guarded choice)	|	$g + g$	(choice)
|	$s \mid s$	(parallel composition)			
|	$\{\!\{s\}\!\}$	(protection)			
|	$[e]\, s$	(delimitation)			
|	$* s$	(replication)			

Table 7. COWS syntax

$\{\!\{\mathbf{0}\}\!\}$	$\equiv$	$\mathbf{0}$	$[k]\,\mathbf{0}$	$\equiv$	$\mathbf{0}$	
$\{\!\{\{\!\{s\}\!\}\}\!\}$	$\equiv$	$\{\!\{s\}\!\}$	$[k_1]\,[k_2]\,s$	$\equiv$	$[k_2]\,[k_1]\,s$	
$\{\!\{[e]\, s\}\!\}$	$\equiv$	$[e]\,\{\!\{s\}\!\}$	$s_1 \mid [k]\, s_2$	$\equiv$	$[k]\,(s_1 \mid s_2)$	if $k \notin \mathrm{fk}(s_1) \cup \mathrm{fk}(s_2)$

Table 8. COWS structural congruence (additional laws)

5.1 Syntax

The syntax of COWS is presented in Table 7 (the new constructs are highlighted by a gray background). In COWS, besides the sets of values and variables, we also use the set of *(killer) labels* (ranged over by $k, k', \ldots$). Notably, expressions do not include killer labels that, hence, are *non-communicable* values. This way the scope of killer labels cannot be dynamically extended and the activities whose termination would be forced by execution of a kill can be statically determined.

We still use w to range over values and variables, u to range over names and variables, while we use e to range over *elements*, namely killer labels, names and variables. Delimitation now is a binder also for killer labels. $\mathrm{fe}(t)$ denotes the set of free elements in t, and $\mathrm{fk}(t)$ denotes the set of free killer labels in t. A closed service is a COWS term without free variables and killer labels.

5.2 Operational semantics

The structural congruence $\equiv$ for COWS, besides the laws in Table 2, additionally includes the laws in Tables 8. Notably, the last law of Table 8 prevents extending the scope of a killer label k when it is free in s_1 or s_2 (this avoids involving s_1 in the effect of a kill activity inside s_2 and is essential to statically determine which activities can be terminated by a kill). Thus, this law can be used to garbage-collect killer labels, e.g. $[k]\,\mathtt{n}!\bar{\epsilon} \equiv [k]\,(\mathtt{n}!\bar{\epsilon} \mid \mathbf{0}) \equiv \mathtt{n}!\bar{\epsilon} \mid [k]\,\mathbf{0} \equiv \mathtt{n}!\bar{\epsilon} \mid \mathbf{0} \equiv \mathtt{n}!\bar{\epsilon}$.

To define the labelled transition relation, we need two new auxiliary functions. The function $\mathit{halt}(\_)$ takes a service s as an argument and returns the service obtained by only retaining the protected activities inside s. $\mathit{halt}(\_)$ is defined inductively on the syntax of services. The most significant case is $\mathit{halt}(\{\!\{s\}\!\}) = \{\!\{s\}\!\}$. In the other cases, $\mathit{halt}(\_)$ returns $\mathbf{0}$, except for parallel composition, delimitation and replication operators, for which it acts as an homomorphism.

$\mathrm{noKill}(s, e) = \mathbf{true}$ if $\mathrm{fk}(e) = \emptyset$	$\mathrm{noKill}(s \mid s', k) = \mathrm{noKill}(s, k) \wedge \mathrm{noKill}(s', k)$
$\mathrm{noKill}(\mathbf{kill}(k), k) = \mathbf{false}$	$\mathrm{noKill}([e]\, s, k) = \mathrm{noKill}(s, k)$ if $e \neq k$
$\mathrm{noKill}(\mathbf{kill}(k'), k) = \mathbf{true}$ if $k \neq k'$	$\mathrm{noKill}([k]\, s, k) = \mathbf{true}$
$\mathrm{noKill}(\mathrm{u}!\bar{\epsilon}, k) = \mathrm{noKill}(g, k) = \mathbf{true}$	$\mathrm{noKill}(\{\!|s|\!\}, k) = \mathrm{noKill}(*\, s, k) = \mathrm{noKill}(s, k)$

Table 9. There are no active $\mathbf{kill}(k)$

$$\mathbf{kill}(k) \xrightarrow{k} \mathbf{0} \;\; (kill) \qquad \frac{s_1 \xrightarrow{\alpha} s_1' \;\; \alpha \neq k, \mathrm{n}\sigma\ell\bar{v}}{s_1 \mid s_2 \xrightarrow{\alpha} s_1' \mid s_2} \;\; (par_3) \qquad \frac{s_1 \xrightarrow{k} s_1'}{s_1 \mid s_2 \xrightarrow{k} s_1' \mid halt(s_2)} \;\; (par_{kill})$$

$$\frac{s \xrightarrow{k} s'}{[k]\, s \xrightarrow{\dagger} [k]\, s'} \;\; (del_{kill\,1}) \qquad \frac{s \xrightarrow{k} s' \quad k \neq e}{[e]\, s \xrightarrow{k} [e]\, s'} \;\; (del_{kill\,2}) \qquad \frac{s \xrightarrow{\dagger} s'}{[e]\, s \xrightarrow{\dagger} [e]\, s'} \;\; (del_{kill\,3})$$

$$\frac{s \xrightarrow{\alpha} s' \quad e \notin (\mathrm{e}(\alpha) \cup \mathrm{ce}(\alpha)) \quad \alpha \neq k, \dagger \quad \mathrm{noKill}(s, e)}{[e]\, s \xrightarrow{\alpha} [e]\, s'} \;\; (del_3) \qquad \frac{s \xrightarrow{\alpha} s'}{\{\!|s|\!\} \xrightarrow{\alpha} \{\!|s'|\!\}} \;\; (prot)$$

Table 10. COWS operational semantics (additional rules)

$$halt(\mathbf{kill}(k)) = halt(\mathrm{u}!\bar{\epsilon}) = halt(g) = \mathbf{0} \qquad halt(\{\!|s|\!\}) = \{\!|s|\!\}$$

$$halt(s_1 \mid s_2) = halt(s_1) \mid halt(s_2) \qquad halt([e]\, s) = [e]\, halt(s) \qquad halt(*\, s) = *\, halt(s)$$

Then, in Table 9, we inductively define the predicate $\mathrm{noKill}(s, e)$, that holds true if either e is not a killer label or $e = k$ and s cannot immediately perform a free kill activity $\mathbf{kill}(k)$. Moreover, the predicate $\mathrm{noConf}(s, \mathrm{n}, \bar{v}, \ell)$, defined for μCOWS by the rules in Table 6, is extended to COWS by adding the following rules:

$$\mathrm{noConf}(\mathbf{kill}(k), \mathrm{n}, \bar{v}, \ell) = \mathbf{true} \qquad \mathrm{noConf}(\{\!|s|\!\}, \mathrm{n}, \bar{v}, \ell) = \mathrm{noConf}(s, \mathrm{n}, \bar{v}, \ell)$$

$$\mathrm{noConf}([e]\, s, \mathrm{n}, \bar{v}, \ell) = \begin{cases} \mathrm{noConf}(s, \mathrm{n}, \bar{v}, \ell) & \text{if } e \notin \mathrm{n} \\ \mathbf{true} & \text{otherwise} \end{cases}$$

The labelled transition relation $\xrightarrow{\alpha}$ is the least relation over services induced by the rules in Tables 4, 5 and 10, where rule *(del$_3$)* replaces *(del)* and *(del$_2$)*, rule *(com$_2$)* replaces *(com)*, and rule *(par$_3$)* replaces *(par)* and *(par$_2$)*. Labels are now generated by the following grammar:

$$\alpha \;::=\; \mathrm{n} \lhd [\bar{n}]\, \bar{v} \;\mid\; \mathrm{n} \rhd [\bar{x}]\, \bar{w} \;\mid\; \sigma \;\mid\; \mathrm{n}\sigma\ell\bar{v} \;\mid\; k \;\mid\; \dagger$$

The meaning of the new labels is as follows: k denotes execution of a request for terminating a term from within the delimitation $[k]$, and $\dagger$ denotes a computational step corresponding to taking place of forced termination. In the sequel, we use $\mathrm{e}(\alpha)$ to denote the set of elements occurring in α (it is defined similarly to $\mathrm{u}(\alpha)$).

Let us now comment on the operational rules. Activity $\mathbf{kill}(k)$ forces termination of all unprotected parallel activities (rules *(kill)* and *(par$_{kill}$)*) inside an enclosing $[k]$, that stops the killing effect by turning the transition label k into $\dagger$ (rule *(del$_{kill\,1}$)*). Existence of such delimitation is ensured by the assumption that the semantics is only defined for

closed services. Critical activities can be protected from killing by putting them into a protection $\{\!|\_|\!\}$; this way, $\{\!|s|\!\}$ behaves like s (rule *(prot)*). Similarly, $[e]\,s$ behaves like s (rule (del_3)), except when the transition label α contains e, in which case α must correspond either to a communication assigning a value to e (rules (del_{com}) and $(del_{com\,2})$) or to a kill activity for e (rule $(del_{kill\,1})$), or when a free kill activity for e is active in s, in which case only actions corresponding to kill activities can be executed (rules $(del_{kill\,2})$ and $(del_{kill\,3})$), that also apply when the third premise of (del_3) does not hold, i.e. $\alpha = k$ or $\alpha = \dagger$). This means that kill activities are executed *eagerly* with respect to the activities enclosed within the delimitation of the corresponding killer label. Execution of parallel services is interleaved (rule (par_3)), but when a kill activity or a communication is performed. Indeed, the former must trigger termination of all parallel services (according to rule (par_{kill})), while the latter must ensure that the receive activity with greater priority progresses (rules (com_2) and (par_{com})).

5.3 Observational semantics

When considering observational semantics for COWS we soon discover that $\sim_\mu$ is not preserved by those contexts forcing termination of the activities in the hole. For example, $\emptyset \sim_\mu \{\!|\emptyset|\!\}$ trivially holds. However, the COWS context $[k]\,(\mathbf{kill}(k) \mid [\![\cdot]\!])$ can tell the two terms apart. Indeed, $[k]\,(\mathbf{kill}(k) \mid \emptyset) \sim_\mu [k]\,(\mathbf{kill}(k) \mid \{\!|\emptyset|\!\})$ does not hold since, after execution of the kill activity (that has highest priority), we would get $\mathbf{0} \sim_\mu \{\!|\emptyset|\!\}$ which is trivially false. Therefore, $\sim_\mu$ is not a congruence for COWS.

Strong open barbed bisimilarity. Open barbed bisimilarity is by definition closed under all contexts and its definition only needs to be tuned for considering also †-transitions in the 'Computation closure'.

Definition 11 (Open barbed bisimilarity). *A symmetric binary relation $\mathcal{R}$ on* COWS *closed terms is an* open barbed bisimulation *if whenever $s_1 \mathcal{R} s_2$ the following holds:*

(Barb preservation) *if $s_1 \downarrow_{\mathtt{n}}$ then $s_2 \downarrow_{\mathtt{n}}$;*

(Computation closure) *if $s_1 \xrightarrow{\emptyset} s_1'$, $s_1 \xrightarrow{\dagger} s_1'$, $s_1 \xrightarrow{\mathtt{n}\,\emptyset\,\ell\,\bar{v}} s_1'$ respectively, then there exists s_2' such that $s_2 \xrightarrow{\emptyset} s_2'$, $s_2 \xrightarrow{\dagger} s_2'$, $s_2 \xrightarrow{\mathtt{n}\,\emptyset\,\ell\,\bar{v}} s_2'$ or $\ell = |\bar{v}| \wedge s_2 \xrightarrow{\emptyset} s_2'$ respectively, and $s_1' \mathcal{R} s_2'$;*

(Context closure) $\mathbb{C}[\![s_1]\!]\,\mathcal{R}\,\mathbb{C}[\![s_2]\!]$*, for every closed context* $\mathbb{C}$*.*

Two closed terms s_1 and s_2 are open barbed bisimilar*, written $s_1 \simeq s_2$, if $s_1 \mathcal{R} s_2$ for some open barbed bisimulation $\mathcal{R}$. $\simeq$ is called open barbed bisimilarity.*

Strong labelled bisimilarity. Labelled bisimilarity must explicitly take care of the terms resulting from application of function *halt*(_) (that gets the same effect as of plunging a term within the context $[k]\,(\mathbf{kill}(k) \mid [\![\cdot]\!])$). It must also consider that if a closed term s performs a transition labelled by $\mathtt{n} \lhd [\bar{m}]\,\bar{v}$, then s contains an invoke of the form $\mathtt{n}!\bar{\epsilon}$, with $[\![\bar{\epsilon}]\!] = \bar{v}$, which can be either protected or not. These differences w.r.t. to Definition 8 are highlighted with a gray background.

Definition 12 (Labelled bisimilarity). *A names-indexed family of relations $\{\mathcal{R}_\mathcal{N}\}_\mathcal{N}$ is a* labelled bisimulation *if $s_1 \mathcal{R}_\mathcal{N} s_2$ then $halt(s_1)\,\mathcal{R}_\mathcal{N}\, halt(s_2)$ and if $s_1 \xrightarrow{\alpha} s_1'$, where* $\mathrm{bu}(\alpha)$ *are fresh, then:*

1. *if $\alpha = \mathtt{n} \rhd [\bar{x}]\,\bar{w}$ then one of the following holds:*
   (a) $\exists\, s_2' \,:\, s_2 \xrightarrow{\mathtt{n}\rhd[\bar{x}]\,\bar{w}} s_2'$ *and*
   $\forall\, \bar{v}$ *s.t.* $\mathcal{M}(\bar{x},\bar{v}) = \sigma$ *and* $\mathrm{noConf}(s_2, \mathtt{n}, \bar{w}\cdot\sigma, |\bar{x}|) \,:\, s_1'\cdot\sigma\, \mathcal{R}_\mathcal{N}\, s_2'\cdot\sigma$
   (b) $|\bar{x}|{=}|\bar{w}|$ *and* $\exists\, s_2' \,:\, s_2 \xrightarrow{\emptyset} s_2'$ *and* $\forall\, \bar{v}$ *s.t.* $\mathcal{M}(\bar{x},\bar{v}) = \sigma$ *and*
   $\mathrm{noConf}(s_2, \mathtt{n}, \bar{w}\cdot\sigma, |\bar{x}|) \,:\, s_1'\cdot\sigma\, \mathcal{R}_\mathcal{N}\, (s_2' \mid \mathtt{n}!\bar{v})$ *or* $s_1'\cdot\sigma\, \mathcal{R}_\mathcal{N}\, (s_2' \mid \{\!|\mathtt{n}!\bar{v}|\!\})$
2. *if $\alpha = \mathtt{n}\,\emptyset\,\ell\,\bar{v}$ where $\ell = |\bar{v}|$ then one of the following holds:*
   (a) $\exists\, s_2' \,:\, s_2 \xrightarrow{\mathtt{n}\,\emptyset\,\ell\,\bar{v}} s_2'$ *and* $s_1'\,\mathcal{R}_\mathcal{N}\, s_2'$ (b) $\exists\, s_2' \,:\, s_2 \xrightarrow{\emptyset} s_2'$ *and* $s_1'\,\mathcal{R}_\mathcal{N}\, s_2'$
3. *if $\alpha = \mathtt{n} \lhd [\bar{n}]\,\bar{v}$ where $\mathtt{n} \notin \mathcal{N}$ then $\exists\, s_2' \,:\, s_2 \xrightarrow{\mathtt{n}\lhd[\bar{n}]\,\bar{v}} s_2'$ and $s_1'\,\mathcal{R}_{\mathcal{N}\cup\bar{n}}\, s_2'$*
4. *if $\alpha = \emptyset$, $\alpha = \dagger$ or $\alpha = \mathtt{n}\,\emptyset\,\ell\,\bar{v}$, where $\ell \neq |\bar{v}|$, then $\exists\, s_2' \,:\, s_2 \xrightarrow{\alpha} s_2'$ and $s_1'\,\mathcal{R}_\mathcal{N}\, s_2'$*

*Two closed terms s_1 and s_2 are $\mathcal{N}$-*bisimilar*, written $s_1 \sim^\mathcal{N} s_2$, if $s_1 \mathcal{R}_\mathcal{N} s_2$ for some $\mathcal{R}_\mathcal{N}$ in a labelled bisimulation. They are* labelled bisimilar*, written $s_1 \sim s_2$, if they are $\emptyset$-bisimilar. $\sim^\mathcal{N}$ is called $\mathcal{N}$-bisimilarity, while $\sim$ is called labelled bisimilarity.*

Notably, *halt*-closure takes into account execution of outer kill activities (i.e. kills performed by contexts), while the inner ones that are active in the considered terms are taken into account by clause 4.

Remark 2 (On computational steps). COWS supports three different kinds of computational steps (labelled by $\emptyset$, $\mathtt{n}\,\emptyset\,\ell\,\bar{v}$, or $\dagger$) and they are dealt with separately because the priority of forced termination over communication permits to observe the kind of computational steps that take place. Indeed, let τ denote any computational step, and suppose to weaken computation closure in Definition 11 and clause 4 of Definition 12 by only requiring that if $s_1 \xrightarrow{\tau}$ then $s_2 \xrightarrow{\tau}$. We would have that $[k]\,\mathbf{kill}(k) \not\sim \emptyset$ while $[k]\,\mathbf{kill}(k) \sim \emptyset$. In fact, the above terms are easily distinguished by the context $[k']\,(\mathbf{kill}(k') \mid [\![\cdot]\!])$, because the term obtained by filling the context with the term on the left can perform the following transitions

$$[k']\,(\mathbf{kill}(k') \mid [k]\,\mathbf{kill}(k)) \xrightarrow{\tau} [k']\,\mathbf{kill}(k') \xrightarrow{\tau} \mathbf{0}$$

while, due to priority of kill over communication, the other term can only reply as follows

$$[k']\,(\mathbf{kill}(k') \mid \emptyset) \xrightarrow{\tau} \mathbf{0} \not\xrightarrow{\tau}$$

This means that $[k']\,(\mathbf{kill}(k') \mid [k]\,\mathbf{kill}(k)) \not\sim [k']\,(\mathbf{kill}(k') \mid \emptyset)$, and hence $[k]\,\mathbf{kill}(k) \not\sim \emptyset$. Moreover, it would hold that $[k']\,(\mathbf{kill}(k') \mid [k]\,\mathbf{kill}(k)) \not\sim [k']\,(\mathbf{kill}(k') \mid \emptyset)$, which also implies that $\sim$ would not be a congruence for COWS.

Thus, to enable $\sim$ to tell $[k]\,\mathbf{kill}(k)$ and $\emptyset$ apart, we must allow $\sim$ to distinguish computational steps (as in clause 4 of Definition 12). This way, we also have that the following holds:

$$\{\!|\,[k]\,\mathbf{kill}(k)\,|\!\} \not\sim \{\!|\,\emptyset\,|\!\}$$

However, they have the same barbs and no context can take them apart. Thus,

$$\{\!|\, [k]\, \mathbf{kill}(k) \,|\!\} \;\simeq\; \{\!|\, \emptyset \,|\!\}$$

Therefore, to have a sound and complete characterisation of $\simeq$ in terms of $\sim$, we must allow $\simeq$ to distinguish computational steps (as in the computation closure required by Definition 11).

We can now prove that labelled bisimilarity is a congruence for COWS.

Definition 13. *Let s_1 and s_2 be two* COWS *term containing free variables $\bar{x}$ at most. Then $s_1 \sim^{\mathcal{N}} s_2$ if, for all values $\bar{v}$ such that $|\bar{x}|=|\bar{v}|$, $s_1 \cdot \{\bar{x} \mapsto \bar{v}\} \sim^{\mathcal{N}} s_2 \cdot \{\bar{x} \mapsto \bar{v}\}$.*

Theorem 7. *$\sim$ is a congruence for* COWS.

Proof (sketch). We shall prove that, given two COWS closed terms s_1 and s_2, if $s_1 \sim s_2$ then $\mathbb{C}[\![s_1]\!] \sim \mathbb{C}[\![s_2]\!]$ for every context $\mathbb{C}$. The proof is by induction on the structure of the context $\mathbb{C}$. □

We prove now that barbed bisimilarity and labelled bisimilarity coincide.

Theorem 8 (Soundness of $\sim$ w.r.t. $\simeq$). *Given two* COWS *closed terms s_1 and s_2, if $s_1 \sim s_2$ then $s_1 \simeq s_2$.*

Proof. The proof proceeds as the proof of Theorem 5, by also exploiting the fact that, by Definition 12, if $s_1 \xrightarrow{\dagger} s_1'$ then $s_2 \xrightarrow{\dagger} s_2'$ and $s_1' \sim s_2'$, for some s_2'. □

Theorem 9 (Completeness of $\sim$ w.r.t. $\simeq$). *Given two* COWS *closed terms s_1 and s_2, if $s_1 \simeq s_2$ then $s_1 \sim s_2$.*

Proof. The proof proceeds as the proof of Theorem 6, by also exploiting the fact that, by Definition 11, if $s_1 \xrightarrow{\dagger} s_1'$ then $s_2 \xrightarrow{\dagger} s_2'$ and $s_1' \simeq s_2'$, for some s_2'. □

Corollary 3. *$\sim$ and $\simeq$ coincide.*

Proof. Directly from Theorems 8 and 9. □

Weak open barbed and labelled bisimilarities. Extension to the weak case is standard. Let τ denote any computational step, i.e. either $\emptyset$ or $\dagger$. With respect to the strong case, and to the analogous extension for μCOWS^m and μCOWS, we relax the requirement for the computational steps, by merely requiring each τ to be matched by zero or more τ. Specifically, weak transitions are defined as follows: $\Longrightarrow$ means $(\xrightarrow{\tau})^*$, i.e. zero or more τ-transitions; $\overset{\alpha}{\Longrightarrow}$ means $\Longrightarrow\xrightarrow{\alpha}\Longrightarrow$; $\overset{\hat{\alpha}}{\Longrightarrow}$ means $\overset{\alpha}{\Longrightarrow}$ if $\alpha \neq \tau$ and $\Longrightarrow$ if $\alpha = \tau$. Thus, we obtain the following definition of weak bisimulations.

Weak open barbed bisimilarity, namely $\cong$, is obtained by replacing $s \downarrow_\mu$ with $s \Downarrow_\mu$, $\xrightarrow{\emptyset}$ and $\xrightarrow{\dagger}$ with $\Longrightarrow$, and $\xrightarrow{\mathrm{n}\emptyset\ell\bar{v}}$ with $\overset{\mathrm{n}\emptyset\ell\bar{v}}{\Longrightarrow}$ in Definition 11.

Definition 14 (Weak labelled bisimilarity). *A names-indexed family of relations* $\{\mathcal{R}_\mathcal{N}\}_\mathcal{N}$ *is a* weak labelled bisimulation *if* $s_1 \mathcal{R}_\mathcal{N} s_2$ *then* $halt(s_1)\,\mathcal{R}_\mathcal{N}\, halt(s_2)$ *and if* $s_1 \xrightarrow{\alpha} s_1'$, *where* $\mathrm{bu}(\alpha)$ *are fresh, then:*

1. *if* $\alpha = \mathtt{n} \triangleright [\bar{x}]\,\bar{w}$ *then one of the following holds:*
   (a) $\exists\, s_2'' \;:\; s_2 \Longrightarrow \xrightarrow{\mathtt{n}\triangleright[\bar{x}]\,\bar{w}} s_2''$ *and* $\forall\, \bar{v}$ *s.t.* $\mathcal{M}(\bar{x},\bar{v}) = \sigma$ *and* $\mathrm{noConf}(s_2, \mathtt{n}, \bar{w}\cdot\sigma, |\,\bar{x}\,|)$
   $\exists\, s_2' \;:\; s_2''\cdot\sigma \Longrightarrow s_2'$ *and* $s_1'\cdot\sigma\,\mathcal{R}_\mathcal{N}\, s_2'$

   (b) $|\,\bar{x}\,| = |\,\bar{w}\,|$ *and* $\exists\, s_2' \;:\; s_2 \Longrightarrow s_2'$ *and*
   $\forall\, \bar{v}$ *s.t.* $\mathcal{M}(\bar{x},\bar{v}) = \sigma$ *and* $\mathrm{noConf}(s_2, \mathtt{n}, \bar{w}\cdot\sigma, |\,\bar{x}\,|) \;:\; s_1'\cdot\sigma\,\mathcal{R}_\mathcal{N}\,(s_2' \mid \mathtt{n}!\bar{v})$
   or $s_1'\cdot\sigma\,\mathcal{R}_\mathcal{N}\,(s_2' \mid \{\!|\mathtt{n}!\bar{v}|\!\})$
2. *if* $\alpha = \mathtt{n}\,\emptyset\,\ell\,\bar{v}$ *where* $\ell = |\,\bar{v}\,|$ *then one of the following holds:*
   (a) $\exists\, s_2' \;:\; s_2 \xRightarrow{\mathtt{n}\,\emptyset\,\ell\,\bar{v}} s_2'$ *and* $s_1'\,\mathcal{R}_\mathcal{N}\, s_2'$ (b) $\exists\, s_2' \;:\; s_2 \Longrightarrow s_2'$ *and* $s_1'\,\mathcal{R}_\mathcal{N}\, s_2'$
3. *if* $\alpha = \mathtt{n} \triangleleft [\bar{n}]\,\bar{v}$ *where* $\mathtt{n} \notin \mathcal{N}$ *then* $\exists\, s_2' \;:\; s_2 \xRightarrow{\mathtt{n}\triangleleft[\bar{n}]\,\bar{v}} s_2'$ *and* $s_1'\,\mathcal{R}_{\mathcal{N}\cup\bar{n}}\, s_2'$
4. *if* $\alpha = \emptyset$, $\alpha = \dagger$ *or* $\alpha = \mathtt{n}\,\emptyset\,\ell\,\bar{v}$, *where* $\ell \neq |\,\bar{v}\,|$, *then* $\exists\, s_2' \;:\; s_2 \xRightarrow{\hat{\alpha}} s_2'$ *and* $s_1'\,\mathcal{R}_\mathcal{N}\, s_2'$

Two closed terms s_1 *and* s_2 *are* weak $\mathcal{N}$-bisimilar, *written* $s_1 \approx^\mathcal{N} s_2$, *if* $s_1\mathcal{R}_\mathcal{N} s_2$ *for some* $\mathcal{R}_\mathcal{N}$ *in a weak labelled bisimulation. They are* weak labelled bisimilar, *written* $s_1 \approx s_2$, *if they are weak* $\emptyset$*-bisimilar.* $\approx^\mathcal{N}$ *is called weak* $\mathcal{N}$*-bisimilarity, while* $\approx$ *is called weak labelled bisimilarity.*

Again, results of congruence and coincidence hold.

5.4 Analysing the 'Morra game' scenario

We now study the relationship between the high- and low-level specifications of the Morra service introduced in Section 2. We can prove that (1) $\not\approx$ (2). Indeed, (1) can perform transitions $\xrightarrow{evens \bullet throw\triangleright[x_{id},y_p,y_{num}]\,\langle x_{id},y_p,y_{num}\rangle}$ and $\xrightarrow{odds \bullet throw\triangleright[x_p,x_{num}]\,\langle first,x_p,x_{num}\rangle}$ and, because of application of substitutions $\{x_{id} \mapsto first, y_p \mapsto cbB, y_{num} \mapsto 1\}$ and $\{x_p \mapsto cbA, x_{num} \mapsto 2\}$, evolve to

$$(\, cbA \bullet res!\langle first, win(2,1,1)\rangle \;\mid\; cbB \bullet res!\langle first, win(2,1,0)\rangle \,)$$

Instead, (2) can properly simulate the above transitions but it can only evolve to

$$\{\!|\, cbA \bullet res!\langle first, w\rangle \;\mid\; cbB \bullet res!\langle first, l\rangle \,|\!\}$$

Of course, the latter term behaves differently from the former one in presence of kill activities. In fact, given the context $\mathbb{C} \triangleq [k']\,(\,[\mathtt{n}]\,(\mathtt{n}!\langle\rangle \mid \mathtt{n}?\langle\rangle.\,\mathbf{kill}(k'))\mid [\![\cdot]\!]\,)$, we have that $\mathbb{C}[\![(1)]\!] \not\approx \mathbb{C}[\![(2)]\!]$.

If we modify the last two invoke activities in the high-level specification (1) as follows:

$$\begin{array}{l} *\,[x_{id}, x_p, x_{num}, y_p, y_{num}] \\ \quad (\, odds \bullet throw?\langle x_{id}, x_p, x_{num}\rangle \;\mid\; evens \bullet throw?\langle x_{id}, y_p, y_{num}\rangle \\ \qquad \mid\; \{\!|\, x_p \bullet res!\langle x_{id}, win(x_{num}, y_{num}, 1)\rangle \;\mid\; y_p \bullet res!\langle x_{id}, win(x_{num}, y_{num}, 0)\rangle \,|\!\}\,) \end{array} \tag{6}$$

the problem persists, because (6) after the first two transitions always provides a response, while (2) could fail to provide a response in presence of kill activities. Instead, by replacing M in (2) by

$$[x_{id}, x_p, x_{num}, y_p, y_{num}]$$
$$(\, odds \bullet throw?\langle x_{id}, x_p, x_{num}\rangle \mid evens \bullet throw?\langle x_{id}, y_p, y_{num}\rangle \mid \{\!|\, [k]\,(\ldots)\,|\!\}\,)$$

we can equate (6) with the so obtained low-level specification.

We can also prove that M is congruent to the following variant:

$$\begin{array}{l}[x_{id}, x_p, x_{num}, y_p, y_{num}]\\ (\, odds \bullet throw?\langle x_{id}, x_p, x_{num}\rangle \mid evens \bullet throw?\langle x_{id}, y_p, y_{num}\rangle\\ \quad \mid m \bullet req2f!\langle x_{id}, x_{num}, y_{num}\rangle \mid m \bullet req5f!\langle x_{id}, x_{num}, y_{num}\rangle\\ \quad \mid [x_o, x_e]\,(\, m \bullet resp2f?\langle x_{id}, x_o, x_e\rangle.\, \{\!|\, x_p \bullet res!\langle x_{id}, x_o\rangle \mid y_p \bullet res!\langle x_{id}, x_e\rangle |\!\}\\ \qquad\qquad + m \bullet resp5f?\langle x_{id}, x_o, x_e\rangle.\, \{\!|\, x_p \bullet res!\langle x_{id}, x_o\rangle \mid y_p \bullet res!\langle x_{id}, x_e\rangle |\!\}\,)\,)\end{array}$$

This somehow suggests us that we can use kill, delimitation and protection to model choice among receive activities that prefix protected terms, like in the following case

$$[\bar{x}]\,(\, \mathrm{n}?\bar{w}.\, \{\!|\, s |\!\} + \mathrm{n}'?\bar{w}'.\, \{\!|\, s' |\!\}\,) \;\approx\; [k, \bar{x}]\,(\, \mathrm{n}?\bar{w}.\, (\mathbf{kill}(k) \mid \{\!|\, s |\!\}) \mid \mathrm{n}'?\bar{w}'.\, (\mathbf{kill}(k) \mid \{\!|\, s' |\!\})\,)$$

6 Concluding remarks and related work

We have investigated the impact of COWS's dynamic priority mechanisms combined with local pre-emption on the definitions of semantic theories for SOC systems. We have introduced natural notions of strong and weak open barbed bisimilarities for COWS, and then proved their coincidence with more manageable characterisations in terms of labelled bisimilarities. We have also demonstrated our approach through the specification and the analysis of an illustrative example.

We leave for future work the identification of appropriate sets of sound and general equational laws that can facilitate the task of analysing SOC systems through the semantic theories introduced in this paper. In fact, at a certain level of abstraction, WS-BPEL applications can be modelled using COWS, thus we hope eventually to be able to verify them. We also plan to develop efficient symbolic characterisations of the labelled bisimilarities over a new symbolic operational semantics for COWS introduced in [42].

The birth of the SOC paradigm has caused the development of new technologies and has stimulated foundational research. Many new process calculi have been designed (see e.g. [13, 31, 26, 6, 21, 4, 14, 30, 7, 11, 44]), addressing one aspect or another of SOC and aiming at assessing the adequacy of diverse sets of primitives w.r.t. modelling, combining and analysing SOC applications. COWS borrows a number of features from well-known 'traditional' process calculi: asynchronous [8, 28, 3] and polyadic communication [39], polyadic synchronization [15], and localised channels [34] have been deeply studied in the setting of π-calculus [37], as well as input-guarded choice (that has been used in many presentations of π-calculus); global scoping and non-binding input are inspired by fusion calculus [41]; pattern matching-based communication has been used in, e.g., Klaim [17] and Xpi [2]; distinction between variables and values is present in, e.g., value-passing CCS [35], applied π-calculus [1] and Dπ [27].

The most distinctive feature of COWS is, however, the parallel operator that takes priority of actions into account, where actions have assigned priority values that can dynamically change and have a scope. In [33], we have largely discussed the relevance of this feature to deal with correlation, and service instantiation and termination. Many process calculi with priority have been proposed in the literature; a comprehensive survey, with terminology and classification of different approaches, can be found in [16]. In previous proposals, dynamic priorities are basically used to model scheduling approaches and real-time aspects (see e.g. [5, 10, 20]) while in COWS they are used for coordination, other than for orchestration, purposes. For example, in the service *5F* of Section 2 they enable implementing a sort of 'default' behaviour, that returns *err* when a throw is not admissible. To the best of our knowledge (see also [16]), the interplay between dynamic priorities and local pre-emption, and their impact on semantic theories of processes have never been explored before.

In [33] we have also shown how COWS's constructs dealing with termination can be used to model fault and compensation handling à la WS-BPEL, and to encode features and primitives of other calculi for SOC, e.g. the construct *where* of Orc [40] and session closure of SCC [6]. A termination construct similar to COWS's **kill** activity has been introduced in [25] in the setting of a distributed π-calculus, where it is used to model the failure of a node (and, hence, the termination of all the processes running therein). Instead, COWS's **kill** activity (in conjunction with protection and delimitation) is more flexible since it permits terminating parallel activities in a more selective way. Indeed, the construct proposed in [25] can be easily rendered in COWS.

In some calculi for SOC, strong or weak notions of labelled bisimilarity have been used for proving existence of normal forms for services [44], for program transformation [22], for checking conformance of a calculus of orchestration to a calculus of choreography [12], and for proving compliance between service implementations and specifications [30, 11]. The latter two mentioned works share our same aims, but consider process calculi that, to link together actions executed as part of the same interaction, rely on *sessions* rather than on correlation data. Behavioural theories based on testing preorders have instead been exploited to check if a service exposing a given contract can play a specific role within a service choreography [9].

COWS's barbed bisimilarities follow the approach of open barbed bisimilarities [29, 43] rather than that of barbed congruence [38], i.e. quantification over contexts occurs recursively inside the definition of bisimilarity. This approach is commonly used in recent works on defining barbed bisimilarities and, then, their labelled characterizations for process calculi with intricate features as e.g. Distributed π-calculus [24] and KLAIM [18]. This because it simplifies the proof of completeness since it allows to choose a context at any computational step. Moreover, for asynchronous π-calculus, it has been proved in [23] that the two approaches coincide. However, since this is not true for (synchronous) π-calculus [43], and since our calculus is not completely asynchronous, it is probably worth considering also the approach used in [38, 3]. We leave this for future investigation.

COWS's labelled bisimilarities strongly recall the bisimilarities for asynchronous π-calculus introduced in [3]. However, COWS's priority mechanisms make some receive actions observable (which leads to a novel notion of observation that refines the purely

asynchronous one), and require specific conditions on the labelled bisimilarities for these to be congruences.

References

1. M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *Proc. of 27th Annual ACM Symposium on Principles of Programming Languages (POPL)*, pages 104–115. ACM Press, 2001.
2. L. Acciai and M. Boreale. Xpi: a typed process calculus for XML messaging. *Science of Computer Programming*, 71(2):110–143, 2008.
3. R.M. Amadio, I. Castellani, and D. Sangiorgi. On Bisimulations for the Asynchronous pi-Calculus. *Theoretical Computer Science*, 195(2):291–324, 1998.
4. M. Bartoletti, P. Degano, and G. Ferrari. Security Issues in Service Composition. In *Proc. of 8th International Conference on Formal Methods for Open Object-Based Distributed Systems (FMOODS)*, volume 4037 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2006.
5. G. Bhat, R. Cleaveland, and G. Lüttgen. A Practical Approach to Implementing Real-Time Semantics. *Annals of Software Engineering*, 7:127–155, 1999.
6. M. Boreale, R. Bruni, L. Caires, R. De Nicola, I. Lanese, M. Loreti, F. Martins, U. Montanari, A. Ravara, D. Sangiorgi, V.T. Vasconcelos, and G. Zavattaro. SCC: a Service Centered Calculus. In *Proc. of 3rd International Workshop on Web Services and Formal Methods (WS-FM))*, volume 4184 of *Lecture Notes in Computer Science*, pages 38–57. Springer, 2006.
7. M. Boreale, R. Bruni, R. De Nicola, and M. Loreti. Sessions and Pipelines for Structured Service Programming. In *Proc. of 10th International Conference on Formal Methods for Open Object-based Distributed Systems (FMOODS)*, volume 5051 of *Lecture Notes in Computer Science*, pages 19–38. Springer, 2008.
8. G. Boudol. Asynchrony and the pi-calculus. Technical Report 1702, INRIA, Sophia Antipolis, 1991.
9. M. Bravetti and G. Zavattaro. Towards a Unifying Theory for Choreography Conformance and Contract Compliance. In *Software Composition*, volume 4829 of *Lecture Notes in Computer Science*, pages 34–50. Springer, 2007.
10. P. Brémond-Grégoire, S.B. Davidson, and I. Lee. CCSR92: Calculus for Communicating Shared Resources with Dynamic Priorities. In *Proc. of 1st North American Process Algebra Workshop (NAPAW)*, Workshops in Computing, pages 65–85. Springer, 1993.
11. R. Bruni, I. Lanese, H.C. Melgratti, and E. Tuosto. Multiparty sessions in SOC. In *Proc. of 10th international conference on Coordination Models and Languages (COORDINATION)*, volume 5052 of *Lecture Notes in Computer Science*, pages 67–82. Springer, 2008.
12. N. Busi, R. Gorrieri, C. Guidi, R. Lucchi, and G. Zavattaro. Choreography and orchestration conformance for system design. In *Proc. of 8th international conference on Coordination Models and Languages (COORDINATION'06)*, volume 4038 of *Lecture Notes in Computer Science*, pages 63–81. Springer, 2006.
13. M.J. Butler, C.A.R. Hoare, and C. Ferreira. A Trace Semantics for Long-Running Transactions. In *25 Years Communicating Sequential Processes*, volume 3525 of *Lecture Notes in Computer Science*, pages 133–150. Springer, 2005.
14. M. Carbone, K. Honda, and N. Yoshida. Structured Communication-Centred Programming for Web Services. In *Proc. of 16th European Symposium on Programming (ESOP)*, volume 4421 of *Lecture Notes in Computer Science*, pages 2–17. Springer, 2007.
15. M. Carbone and S. Maffeis. On the expressive power of polyadic synchronisation in π-calculus. *Nordic Journal of Computing*, 10(2):70–98, 2003.

16. R. Cleaveland, G. Lüttgen, and V. Natarajan. Priorities in process algebra. *Handbook of Process Algebra, chapter 12*, pages 391–424, 2001.
17. R. De Nicola, G. Ferrari, and R. Pugliese. KLAIM: A Kernel Language for Agents Interaction and Mobility. *Transactions on Software Engineering*, 24(5):315–330, 1998.
18. R. De Nicola, D. Gorla, and R. Pugliese. Basic observables for a calculus for global computing. *Information and Computation*, 205(10):1491–1525, 2007.
19. A. Fantechi, S. Gnesi, A. Lapadula, F. Mazzanti, R. Pugliese, and F. Tiezzi. A model checking approach for verifying COWS specifications. In *Proc. of Fundamental Approaches to Software Engineering (FASE)*, volume 4961 of *Lecture Notes in Computer Science*, pages 230–245. Springer, 2008.
20. H. Fecher. A Real-Time Process Algebra with Open Intervals and Maximal Progress. *Nordic Journal of Computing*, 8(3):346–365, 2001.
21. G. Ferrari, R. Guanciale, and D. Strollo. Event based service coordination over dynamic and heterogeneous networks. In *Proc. of 4th International Conference on Service Oriented Computing (ICSOC)*, volume 4294 of *Lecture Notes in Computer Science*, pages 453–458. Springer, 2006.
22. L.C. Filipe, I. Lanese, V. Vasconcelos, F. Martins, and A. Ravara. Behavioural Theory at Work: Program Transformations in a Service-Centred Calculus. In *Proc. of 10th International Conference on Formal Methods for Open Object-based Distributed Systems (FMOODS)*, volume 5051 of *Lecture Notes in Computer Science*, pages 59–77. Springer, 2008.
23. C. Fournet and G. Gonthier. A Hierarchy of Equivalences for Asynchronous Calculi. In *Proc. of 25th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 1443 of *Lecture Notes in Computer Science*, pages 844–855. Springer, 1998.
24. A. Francalanza and M. Hennessy. A Theory of System Fault Tolerance. In *Proc. of 9th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS)*, volume 3921 of *Lecture Notes in Computer Science*, pages 16–31. Springer, 2006.
25. A. Francalanza and M. Hennessy. A theory for observational fault tolerance. *J. Log. Algebr. Program.*, 73(1-2):22–50, 2007.
26. C. Guidi, R. Lucchi, R. Gorrieri, N. Busi, and G. Zavattaro. SOCK: A Calculus for Service Oriented Computing. In *Proc. of 4th International Conference on Service Oriented Computing (ICSOC)*, volume 4294 of *Lecture Notes in Computer Science*, pages 327–338. Springer, 2006.
27. M. Hennessy and J. Riely. Resource access control in systems of mobile agents. *Information and Computation*, 173(1):82–120, 2002.
28. K. Honda and M. Tokoro. An Object Calculus for Asynchronous Communication. In *Proc. of 5th European Conference on Object-Oriented Programming (ECOOP)*, volume 512 of *Lecture Notes in Computer Science*, pages 133–147. Springer, 1991.
29. K. Honda and N. Yoshida. On Reduction-Based Process Semantics. *Theoretical Computer Science*, 151(2):437–486, 1995.
30. I. Lanese, F. Martins, A. Ravara, and V.T. Vasconcelos. Disciplining Orchestration and Conversation in Service-Oriented Computing. In *Proc, of 5th International Conference on Software Engineering and Formal Methods (SEFM)*, pages 305–314. IEEE Computer Society Press, 2007.
31. C. Laneve and G. Zavattaro. Foundations of Web Transactions. In *Proc. of 8th International Conference on Foundations of Software Science and Computational Structures (FoSSaCS)*, volume 3441 of *Lecture Notes in Computer Science*, pages 282–298. Springer, 2005.
32. A. Lapadula. *Specification and Analysis of Service-Oriented Applications*. PhD Thesis in Computer Science, Dipartimento di Sistemi e Informatica, Università degli Studi di Firenze,

2009. Available at http://rap.dsi.unifi.it/cows/theses/tiezzi_phdthesis.pdf.
33. A. Lapadula, R. Pugliese, and F. Tiezzi. A calculus for orchestration of web services. Technical report, Dipartimento di Sistemi e Informatica, Università di Firenze, 2006. http://rap.dsi.unifi.it/cows/papers/cows-esop07-full.pdf. An extended abstract appeared in *Proc. of ESOP'07*, LNCS 4421, pages 33-47, Springer.
34. M. Merro and D. Sangiorgi. On asynchrony in name-passing calculi. *Mathematical Structures in Computer Science*, 14(5):715–767, 2004.
35. R. Milner. *A Calculus of Communicating Systems.*, volume 92 of *LNCS*. Springer–Verlag, 1980.
36. R. Milner. *Communication and concurrency.* Prentice-Hall, 1989.
37. R. Milner, J. Parrow, and D. Walker. A Calculus of Mobile Processes, I and II. *Information and Computation*, 100(1):1–40, 41–77, 1992.
38. R. Milner and D. Sangiorgi. Barbed Bisimulation. In *Proc. of 19th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 623 of *Lecture Notes in Computer Science*, pages 685–695. Springer, 1992.
39. Robin Milner. The polyadic pi-calculus: a tutorial. Technical report, Logic and Algebra of Specification, 1991.
40. J. Misra and W.R. Cook. Computation Orchestration: A Basis for Wide-Area Computing. *Journal of Software and Systems Modeling*, 6(1):83–110, 2007.
41. J. Parrow and B. Victor. The fusion calculus: Expressiveness and symmetry in mobile processes. In *Proc. of 13th Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 176–185. IEEE Computer Society Press, 1998.
42. R. Pugliese, F. Tiezzi, and N. Yoshida. A Symbolic Semantics for a Calculus for Service-Oriented Computing. In *Proc. of 1st Workshop on Programming Language Approaches to Concurrency and Communication-cEntric Software (PLACES)*, Electronic Notes in Theoretical Computer Science. Elsevier, 2009. To appear.
43. D. Sangiorgi and D. Walker. On Barbed Equivalences in pi-Calculus. In *Proc. of 12th International Conference on Concurrency Theory (CONCUR)*, volume 2154 of *Lecture Notes in Computer Science*, pages 292–304. Springer, 2001.
44. H.T. Vieira, L. Caires, and J. Costa Seco. The Conversation Calculus: A Model of Service-Oriented Computation. In *Proc. of 17th European Symposium on Programming (ESOP)*, volume 4960 of *Lecture Notes in Computer Science*, pages 269–283. Springer, 2008.

A Proofs of main results

A.1 μCOWS^m

Theorem A1 (Theorem 1) *$\sim_m$ is a congruence for μCOWS^m closed terms.*

Proof. We shall prove that, given two μCOWS^m closed terms s_1 and s_2, if $s_1 \sim_m s_2$ then $\mathbb{C}[\![s_1]\!] \sim_m \mathbb{C}[\![s_2]\!]$ for every (possibly open) context $\mathbb{C}$. The proof is by induction on the structure of the context $\mathbb{C}$. The base case, i.e. whenever $\mathbb{C} = [\![\ ]\!]$, is trivial. For the inductive case, we have the following possibilities:

- $\mathbb{C} = \mathtt{n}?\bar{w}.\,\mathbb{D}$.
  By induction, we may assume that $\mathbb{D}[\![s_1]\!] \sim_m \mathbb{D}[\![s_2]\!]$. If $\mathbb{C}$ is a closed context, then $\bar{w} = \bar{v}$, i.e. $\bar{w}$ only contains values, $\mathbb{D}$ is a closed context, and there exists a bisimulation $\mathcal{F}$ such that $\mathbb{D}[\![s_1]\!]\mathcal{R}_\emptyset\mathbb{D}[\![s_2]\!]$ with $\mathcal{R}_\emptyset \in \mathcal{F}$. By Lemma 1, we can prove the thesis by showing that $(\mathcal{F}\backslash\mathcal{R}_\emptyset) \cup \mathcal{R}'_\emptyset$ is a bisimulation up-to $\equiv$, where

  $$\mathcal{R}'_\emptyset = \{(\mathtt{n}?\bar{v}.\,\mathbb{D}[\![s_1]\!], \mathtt{n}?\bar{v}.\,\mathbb{D}[\![s_2]\!])\} \cup \mathcal{R}_\emptyset$$

  Indeed, $\mathcal{R}'_\emptyset$ is *transition closed* because $\mathcal{R}_\emptyset$ is transition closed and

  $$\mathtt{n}?\bar{v}.\,\mathbb{D}[\![s_1]\!] \xrightarrow{\mathtt{n} \rhd \bar{v}} \equiv \mathbb{D}[\![s_1]\!]$$
  $$\mathtt{n}?\bar{v}.\,\mathbb{D}[\![s_2]\!] \xrightarrow{\mathtt{n} \rhd \bar{v}} \equiv \mathbb{D}[\![s_2]\!]$$

  From the hypothesis $\mathbb{D}[\![s_1]\!]\mathcal{R}_\emptyset\mathbb{D}[\![s_2]\!]$, since $\mathcal{R}_\emptyset \subseteq \mathcal{R}'_\emptyset$, we have $\mathbb{D}[\![s_1]\!]\mathcal{R}'_\emptyset\mathbb{D}[\![s_2]\!]$.
  Instead, if $\bar{w}$ and $\mathbb{D}$ contain free variables $\bar{x}$ at most, then the hypothesis $\mathbb{D}[\![s_1]\!] \sim_m \mathbb{D}[\![s_2]\!]$ implies that for all $\bar{v}$ such that $|\ \bar{x}\ | = |\ \bar{v}\ |$ we have $\mathbb{D}'[\![s_1]\!] \sim_m \mathbb{D}'[\![s_2]\!]$ for $\mathbb{D}' = \mathbb{D} \cdot \{\bar{x} \mapsto \bar{v}\}$. Indeed, since s_1 and s_2 are closed terms, $s_1 \cdot \{\bar{x} \mapsto \bar{v}\} = s_1$ and $s_2 \cdot \{\bar{x} \mapsto \bar{v}\} = s_2$. This means that for all $\bar{v}$ there exists a bisimulation $\mathcal{F}'$ such that $\mathbb{D}'[\![s_1]\!]\mathcal{R}''_\emptyset\mathbb{D}'[\![s_2]\!]$ with $\mathcal{R}''_\emptyset \in \mathcal{F}'$. By Lemma 1, we can prove the thesis by showing that $(\mathcal{F}'\backslash\mathcal{R}''_\emptyset) \cup \mathcal{R}'''_\emptyset$ is a bisimulation up-to $\equiv$, where

  $$\mathcal{R}'''_\emptyset = \{(\mathtt{n}?\bar{w}\cdot\{\bar{x} \mapsto \bar{v}\}.\,\mathbb{D}'[\![s_1]\!]\,,\ \mathtt{n}?\bar{w}\cdot\{\bar{x} \mapsto \bar{v}\}.\,\mathbb{D}'[\![s_2]\!])\} \cup \mathcal{R}''_\emptyset$$

  The rest of the proof proceeds as before.
- $\mathbb{C} = \mathbb{G} + g$.
  By induction, we may assume that $\mathbb{G}[\![s_1]\!] \sim_m \mathbb{G}[\![s_2]\!]$. If $\mathbb{G}$ is a closed context, then there exists a bisimulation $\mathcal{F}$ such that $\mathbb{G}[\![s_1]\!]\mathcal{R}_\emptyset\mathbb{G}[\![s_2]\!]$ with $\mathcal{R}_\emptyset \in \mathcal{F}$. By Lemma 1, we can prove the thesis by showing that $(\mathcal{F}\backslash\mathcal{R}_\emptyset) \cup \mathcal{R}'_\emptyset$ is a bisimulation up-to $\equiv$, where

  $$\mathcal{R}'_\emptyset = \{(\mathbb{G}[\![s_1]\!] + g, \mathbb{G}[\![s_2]\!] + g)\} \cup \mathcal{R}_\emptyset \cup Id$$

  and Id is the identity relation. Indeed, if $\mathbb{G}[\![s_1]\!] + g \xrightarrow{\alpha} s$, due to the structure of $\mathbb{G}$ and g, then $\alpha = \mathtt{n} \rhd [\bar{x}]\,\bar{v}$ and there are the following possibilities.
  - $g \xrightarrow{\mathtt{n} \rhd [\bar{x}]\,\bar{v}} s$. Thus, $\mathbb{G}[\![s_2]\!] + g \xrightarrow{\mathtt{n} \rhd [\bar{x}]\,\bar{v}} s$ and $s\,\mathcal{R}'_\emptyset\,s$, since $s\,Id\,s$ and $Id \subseteq \mathcal{R}'_\emptyset$.
  - $\mathbb{G}[\![s_1]\!] \xrightarrow{\mathtt{n} \rhd [\bar{x}]\,\bar{v}} s$. From the hypothesis $\mathbb{G}[\![s_1]\!]\mathcal{R}_\emptyset\mathbb{G}[\![s_2]\!]$, then we have two possibilities:

* There exists s' such that $\mathbb{G}[\![s_2]\!] \xrightarrow{\mathtt{n} \rhd [\bar{x}]\,\bar{v}} s'$ and, for all $\bar{v}$ such that $\mathcal{M}(\bar{x}, \bar{v}) = \sigma$, we have $s \cdot \sigma \mathcal{R}_\emptyset\, s' \cdot \sigma$. Then, $\mathbb{G}[\![s_2]\!] + g \xrightarrow{\mathtt{n} \rhd [\bar{x}]\,\bar{v}} s'$. Since $\mathcal{R}_\emptyset \subseteq \mathcal{R}'_\emptyset$, we get $s \cdot \sigma \mathcal{R}'_\emptyset\, s' \cdot \sigma$.
* There exists s' such that $\mathbb{G}[\![s_2]\!] \xrightarrow{\emptyset} s'$ and, for all $\bar{v}$ such that $\mathcal{M}(\bar{x}, \bar{v}) = \sigma$, we have $s \cdot \sigma \mathcal{R}_\emptyset\, (s' \mid \mathtt{n}!(\bar{w} \cdot \sigma))$. Then, $\mathbb{G}[\![s_2]\!] + g \xrightarrow{\emptyset} s'$ and, since $\mathcal{R}_\emptyset \subseteq \mathcal{R}'_\emptyset$, we get $s \cdot \sigma \mathcal{R}'_\emptyset\, (s' \mid \mathtt{n}!(\bar{w} \cdot \sigma))$.

If $\mathbb{C}$ is an open context, we can proceed similarly to the case $\mathbb{C} = \mathtt{n}?\bar{w}.\mathbb{D}$ when $\mathbb{C}$ contains free variables.

– $\mathbb{C} = g + \mathbb{G}$.
We can proceed as in the case $\mathbb{C} = \mathbb{G} + g$.

– $\mathbb{C} = [x]\,\mathbb{D}$.
If $\mathbb{D}$ is a closed context, then $[x]\,\mathbb{D} \equiv \mathbb{D}$ and, by induction, we immediately conclude. Instead, if $\mathbb{D}$ contains the free variable x at most[2], by induction and since s_1 and s_2 are closed terms, it holds that, for all v, $\mathbb{D}_v[\![s_1]\!] \sim_m \mathbb{D}_v[\![s_2]\!]$, where $\mathbb{D}_v = \mathbb{D} \cdot \{x \mapsto v\}$. This means that, for all v, there exists a bisimulation $\mathcal{F}^v$ such that $\mathbb{D}_v[\![s_1]\!]\, \mathcal{R}^v_\emptyset\, \mathbb{D}_v[\![s_2]\!]$, with $\mathcal{R}^v_\emptyset \in \mathcal{F}^v$. Now, consider the following family of relations:

$$\mathcal{F} = \{\mathcal{R}^{[x]}_{\mathcal{N}} \;:\; \mathcal{N} \text{ is a set of names}\} \cup \bigcup_v \mathcal{F}^v$$

where $\mathcal{R}^{[x]}_{\mathcal{N}} = \{([x]\,s, [x]\,s') \;:\; \forall v \;\; s \cdot \{x \mapsto v\}\, \mathcal{R}^v_{\mathcal{N}}\, s' \cdot \{x \mapsto v\},\ \mathcal{R}^v_{\mathcal{N}} \in \mathcal{F}^v,\ s \text{ and } s' \text{ satisfy } (*)\}$; property $(*)$ states that if $s \xrightarrow{\alpha}$ where $\alpha = \mathtt{n} \rhd [\bar{y}]\,\bar{w}$, with $x \in \bar{w}$, or $\alpha = \sigma \uplus \{x \mapsto v\}$, then $s' \xrightarrow{\alpha}$, and vice versa; $\bigcup_v \mathcal{F}^v = \{\mathcal{R}_{\mathcal{N}} \;:\; \mathcal{N} \text{ is a set of names}, \mathcal{R}^v_{\mathcal{N}} \in \mathcal{F}^v, \mathcal{R}_{\mathcal{N}} = \bigcup_v \mathcal{R}^v_{\mathcal{N}}\}$. The proof proceeds by proving that $\mathcal{F}$ is a bisimulation up-to $\equiv$. Indeed, by letting $s = \mathbb{D}[\![s_1]\!]$ and $s' = \mathbb{D}[\![s_2]\!]$, we obtain the thesis. In fact, $\mathbb{D}_v[\![s_1]\!]\, \mathcal{R}^v_\emptyset\, \mathbb{D}_v[\![s_2]\!]$ for all v, and $\mathbb{D}[\![s_1]\!]$ and $\mathbb{D}[\![s_2]\!]$ satisfy property $(*)$, because x does not occur free in s_1 and s_2 (since s_1 and s_2 are closed terms).

Thus, let us consider a relation $\mathcal{R}^{[x]}_{\mathcal{N}} \in \mathcal{F}$. If $[x]\,s \xrightarrow{\alpha} s''$, we proceed by case analysis on α. The most interesting case is $\alpha = \mathtt{n} \rhd [x, \bar{y}]\,\bar{w}$. By rule *(open$_{rec}$)*, we get that $s \xrightarrow{\mathtt{n} \rhd [\bar{y}]\,\bar{w}} s''$. From this, for all v, $s \cdot \{x \mapsto v\} \xrightarrow{\mathtt{n} \rhd [\bar{y}]\,(\bar{w} \cdot \{x \mapsto v\})} s'' \cdot \{x \mapsto v\}$. By definition of $\mathcal{R}^{[x]}_{\mathcal{N}}$, for all v, we get that $s \cdot \{x \mapsto v\} \mathcal{R}^v_{\mathcal{N}} s' \cdot \{x \mapsto v\}$, where $\mathcal{R}^v_{\mathcal{N}}$ belongs to a bisimulation. Hence, we would have two possibilities: $s' \cdot \{x \mapsto v\} \xrightarrow{\mathtt{n} \rhd [\bar{y}]\,(\bar{w} \cdot \{x \mapsto v\})}$ or $s' \cdot \{x \mapsto v\} \xrightarrow{\emptyset}$. In the former case, because of property $(*)$, we get that $s' \cdot \{x \mapsto v\} \xrightarrow{\mathtt{n} \rhd [\bar{y}]\,(\bar{w} \cdot \{x \mapsto v\})} s''' \cdot \{x \mapsto v\}$ and, for all $\bar{v}'$ s.t. $\mathcal{M}(\bar{y}, \bar{v}') = \sigma$, $s'' \cdot \{x \mapsto v\} \uplus \sigma\, \mathcal{R}^v_{\mathcal{N}}\, s''' \cdot \{x \mapsto v\} \uplus \sigma$, where s''' is such that $s' \xrightarrow{\mathtt{n} \rhd [\bar{y}]\,\bar{w}} s'''$. By rule *(open$_{rec}$)*, we conclude that $[x]\,s' \xrightarrow{\mathtt{n} \rhd [x, \bar{y}]\,\bar{w}} s'''$.

In the latter case, i.e. when $\alpha = \emptyset$, we have two possibilities. In one case, $s \xrightarrow{\emptyset} s'''$ with $s'' \equiv [x]\,s'''$. Then, for all v, $s \cdot \{x \mapsto v\} \xrightarrow{\emptyset} s''' \cdot \{x \mapsto v\}$. By definition of $\mathcal{R}^{[x]}_{\mathcal{N}}$, there exits s'''' such that $s' \cdot \{x \mapsto v\} \xrightarrow{\emptyset} s'''' \cdot \{x \mapsto v\}$ and $s''' \cdot \{x \mapsto v\} \mathcal{R}^v_{\mathcal{N}} s'''' \cdot \{x \mapsto$

[2] The case where $\mathbb{D}$ contains more than one free variable can be dealt with in a similar way.

$v\}$, for some $\mathcal{R}^v_{\mathcal{N}}$ belonging to a bisimulation. Since the communication does not involve x, we get that $s' \xrightarrow{\emptyset} s''''$ and, hence, $[x]\,s' \xrightarrow{\emptyset} [x]\,s''''$. We conclude by noticing that, by definition of $\mathcal{R}^{[x]}_{\mathcal{N}}$, it holds that $[x]\,s'''\mathcal{R}^{[x]}_{\mathcal{N}}\,[x]\,s''''$. The other possibility is that $s \xrightarrow{\{x\mapsto v'\}} s'''$ with $s'' = s''' \cdot \{x \mapsto v'\}$. Then, for all v, $s \xrightarrow{\emptyset} s''' \cdot \{x \mapsto v\}$. By definition of $\mathcal{R}^{[x]}_{\mathcal{N}}$, there exits s'''' such that $s' \cdot \{x \mapsto v\} \xrightarrow{\emptyset} s'''' \cdot \{x \mapsto v\}$ and $s''' \cdot \{x \mapsto v\}\mathcal{R}^v_{\mathcal{N}} s'''' \cdot \{x \mapsto v\}$, for some $\mathcal{R}^v_{\mathcal{N}}$ belonging to a bisimulation. By property $(*)$, $s' \xrightarrow{\{x\mapsto v'\}} s''''$ and, hence, $[x]\,s' \xrightarrow{\emptyset} s'''' \cdot \{x \mapsto v'\}$. Finally, from $s''' \cdot \{x \mapsto v\}\mathcal{R}^v_{\mathcal{N}} s'''' \cdot \{x \mapsto v\}$, by letting $v = v'$ and by definition of $\mathcal{R}^{[x]}_{\mathcal{N}}$, we get that $s''' \cdot \{x \mapsto v'\}\mathcal{R}^{[x]}_{\mathcal{N}}\, s'''' \cdot \{x \mapsto v'\}$.

– $\mathbb{C} = \mathbb{D} \mid s$.
By induction, we may assume that $\mathbb{D}[\![s_1]\!] \sim_m \mathbb{D}[\![s_2]\!]$. If $\mathbb{D}$ is a closed context, then there exists a bisimulation $\mathcal{F}$ such that $\mathbb{D}[\![s_1]\!]\mathcal{R}_\emptyset\mathbb{D}[\![s_2]\!]$ with $\mathcal{R}_\emptyset \in \mathcal{F}$. By Lemma 1, we can prove the thesis by showing that the family of relations

$$\mathcal{F}' = \{\, \mathcal{R}'_{\mathcal{N}'} \;:\; \mathcal{R}_{\mathcal{N}} \in \mathcal{F}\,,\ \mathcal{N} \subseteq \mathcal{N}' \,\}$$

$$\mathcal{R}'_{\mathcal{N}'} = \{\, (\,[\bar{n}]\,(s' \mid s), [\bar{n}]\,(s'' \mid s)\,) \;:\; s'\mathcal{R}_{\mathcal{N}}s''\,, \text{ for some } \bar{n} \text{ and } s \text{ s.t. } \mathcal{N} \cap \mathrm{re}(s) = \emptyset \,\}$$

where $\mathrm{re}(s)$ is the set of the receiving endpoint used in s, is a bisimulation up-to $\equiv$. Notice that for each relation $\mathcal{R}_{\mathcal{N}} \in \mathcal{F}$ there exists a family of relations $\mathcal{R}'_{\mathcal{N}'} \in \mathcal{F}'$ for $\mathcal{N} \subseteq \mathcal{N}'$ such that $\mathcal{R}_{\mathcal{N}} \subseteq \mathcal{R}'_{\mathcal{N}'}$ (in fact, s can be $\mathbf{0}$ and $\bar{n}$ can be empty).

Let us consider a relation $\mathcal{R}'_{\mathcal{N}'} \in \mathcal{F}'$. If $[\bar{n}]\,(s' \mid s) \xrightarrow{\alpha} s'''$, then the proof proceeds by case analysis on α. We only take a look at the cases of bound invocation and communication of private names.

- $\alpha = \mathtt{n} \lhd [\bar{m}]\,\bar{v}$
  We have two possibilities:
  * $s \xrightarrow{\mathtt{n}\lhd[\bar{m}']\,\bar{v}} s_3$, $\bar{m}' \subseteq \bar{m}$, $s''' \equiv [\bar{n}\backslash\bar{m}]\,(s' \mid s_3)$.
    If $\mathtt{n} \in \mathcal{N}'$ then we immediately conclude, because Definition 4 does not impose any requirement in this case. Instead, if $\mathtt{n} \notin \mathcal{N}'$ then $[\bar{n}]\,(s'' \mid s) \xrightarrow{\mathtt{n}\lhd[\bar{m}]\,\bar{v}} [\bar{n}\backslash\bar{m}]\,(s'' \mid s_3)$. Since $s'\mathcal{R}_{\mathcal{N}}s''$, by definition of $\mathcal{F}'$ we get that $[\bar{n}\backslash\bar{m}]\,(s' \mid s_3)\,\mathcal{R}'_{\mathcal{N}'\cup\bar{m}}\,[\bar{n}\backslash\bar{m}]\,(s'' \mid s_3)$.
  * $s' \xrightarrow{\mathtt{n}\lhd[\bar{m}']\,\bar{v}} s_3$, $\bar{m}' \subseteq \bar{m}$, $s''' \equiv [\bar{n}\backslash\bar{m}]\,(s_3 \mid s)$.
    If $\mathtt{n} \in \mathcal{N}'$ then we immediately conclude. Otherwise, since $s'\mathcal{R}_{\mathcal{N}}s''$ for $\mathcal{R}_{\mathcal{N}}$ belonging to the bisimulation $\mathcal{F}$, we get that there exists s_4 such that $s'' \xrightarrow{\mathtt{n}\lhd[\bar{m}']\,\bar{v}} s_4$ and $s_3\,\mathcal{R}_{\mathcal{N}\cup\bar{m}'}\,s_4$. Thus, we have that $[\bar{n}]\,(s'' \mid s) \xrightarrow{\mathtt{n}\lhd[\bar{m}]\,\bar{v}} [\bar{n}\backslash\bar{m}]\,(s_4 \mid s)$. From this and by definition of $\mathcal{F}'$ we get that $[\bar{n}\backslash\bar{m}]\,(s_3 \mid s)\,\mathcal{R}_{\mathcal{N}'\cup\bar{m}}\,[\bar{n}\backslash\bar{m}]\,(s_4 \mid s)$.
- $\alpha = \emptyset$, $s' \xrightarrow{\mathtt{n}\lhd[\bar{m}]\,\bar{v}} s_3$, $s \equiv [\bar{x}]\,s_4 \xrightarrow{\mathtt{n}\rhd[\bar{x}]\,\bar{w}} s_5$, $|\bar{m}| = |\bar{x}|$.
  By rules *(str)*, *(del_{com})* and *(com)*, we get that $(s' \mid s) \equiv [\bar{x}]\,(s' \mid s_4)$, $(s' \mid s_4) \xrightarrow{\{\bar{x}\mapsto\bar{m}\}} (s_3 \mid s_5)$, $\mathcal{M}(\bar{w}, \bar{v}) = \{\bar{x} \mapsto \bar{m}\}$, and $s''' \equiv [\bar{n}, \bar{m}]\,(s_3 \mid s_5 \cdot \{\bar{x} \mapsto \bar{m}\})$. Since $\mathtt{n}$ is a receiving endpoint of s, then $\mathtt{n} \notin \mathcal{N}$. From this and since $s'\mathcal{R}_{\mathcal{N}}s''$ for $\mathcal{R}_{\mathcal{N}}$ belonging to the bisimulation $\mathcal{F}$, we get that there exists s_6 such that

$s'' \xrightarrow{\mathrm{n} \lhd [\bar{m}]\,\bar{v}} s_6$ and $s_3 \mathcal{R}_{\mathcal{N} \cup \bar{m}} s_6$, where $\mathcal{R}_{\mathcal{N} \cup \bar{m}} \in \mathcal{F}$. Hence, $[\bar{n}]\,(s'' \mid s) \xrightarrow{\emptyset} [\bar{n}, \bar{m}]\,(s_6 \mid s_5 \cdot \{\bar{x} \mapsto \bar{m}\})$. Since $s_3 \mathcal{R}_{\mathcal{N} \cup \bar{m}} s_6$, by definition of $\mathcal{F}'$, we conclude that $[\bar{n}, \bar{m}]\,(s_3 \mid s_5 \cdot \{\bar{x} \mapsto \bar{m}\}) \mathcal{R}'_{\mathcal{N}' \cup \bar{m}} [\bar{n}, \bar{m}]\,(s_6 \mid s_5 \cdot \{\bar{x} \mapsto \bar{m}\})$. The generalisation to the case $|\bar{m}| < |\bar{x}|$, where also some non-restricted values are communicated, is trivial.

By letting $s' = \mathbb{D}[\![s_1]\!]$ and $s'' = \mathbb{D}[\![s_2]\!]$, we obtain the thesis. We can generalise to the case where $\mathbb{C}$ is an open context as in the previous cases.

– $\mathbb{C} = s \mid \mathbb{D}$.
We can proceed as in the case $\mathbb{C} = \mathbb{D} \mid s$.

– $\mathbb{C} = [n]\,\mathbb{D}$.
By letting $s = \mathbf{0}$ in the family of relations defined in the case $\mathbb{C} = \mathbb{D} \mid s$, we obtain the thesis.

– $\mathbb{C} = * \mathbb{D}$.
By induction, we may assume that $\mathbb{D}[\![s_1]\!] \sim_m \mathbb{D}[\![s_2]\!]$. If $\mathbb{D}$ is a closed context, then there exists a bisimulation $\mathcal{F}$ such that $\mathbb{D}[\![s_1]\!] \mathcal{R}_\emptyset \mathbb{D}[\![s_2]\!]$ with $\mathcal{R}_\emptyset \in \mathcal{F}$. By Lemma 1, we can prove the thesis by showing that the family of relations

$$\{\, \{\, (* s \mid s'', * s' \mid s'')\ :\ s \mathcal{R}_\mathcal{N} s'\text{, for some } s''\,\}\ :\ \mathcal{R}_\mathcal{N} \in \mathcal{F}\,\}$$

is a bisimulation up-to $\equiv$. The proof proceeds similarly to that for the bisimulation defined in the case $\mathbb{D} \mid s$.
By letting $s = \mathbb{D}[\![s_1]\!]$, $s' = \mathbb{D}[\![s_2]\!]$ and $s'' = \mathbf{0}$, we obtain the thesis. We can generalise to the case where $\mathbb{C}$ is an open context as in the previous cases. □

Theorem A2 (Soundness of $\sim_m$ w.r.t. $\simeq_m$, Theorem 2) *Given two μCOWS^m closed terms s_1 and s_2, if $s_1 \sim_m s_2$ then $s_1 \simeq_m s_2$.*

Proof. By Theorem A1, we have that $\sim_m$ is context closed. Thus, we only need to prove that $\sim_m$ is barb preserving and computation closed.

– *Barb preservation.* Suppose $s_1 \downarrow_\mathrm{n}$. Then, by Definition 1, this means that $s_1 \xrightarrow{\mathrm{n} \lhd [\bar{n}]\,\bar{v}} s'_1$, for some s'_1, $\bar{n}$ and $\bar{v}$. Since $\sim_m = \sim_m^\emptyset$, by Definition 4, we get that $s_2 \xrightarrow{\mathrm{n} \lhd [\bar{n}]\,\bar{v}} s'_2$, for some s'_2. Thus, by Definition 1, $s_2 \downarrow_\mathrm{n}$.
– *Computation closure.* By hypothesis, there exists a labelled bisimulation $\mathcal{F}$ such that $s_1 \mathcal{R}_\emptyset s_2$ with $\mathcal{R}_\emptyset \in \mathcal{F}$. Thus, if $s_1 \xrightarrow{\emptyset} s'_1$, by Definition 4, we get that $s_2 \xrightarrow{\emptyset} s'_2$ and $s'_1 \mathcal{R}_\emptyset s'_2$. This means that $s'_1 \sim_m s'_2$. □

Theorem A3 (Completeness of $\sim_m$ w.r.t. $\simeq_m$, Theorem 3) *Given two μCOWS^m closed terms s_1 and s_2, if $s_1 \simeq_m s_2$ then $s_1 \sim_m s_2$.*

Proof. We define a family of relations $\mathcal{F} = \{\mathcal{R}_\mathcal{N}\ :\ \mathcal{N} \text{ set of names}\}$ such that $\simeq_m$ is included in $\mathcal{R}_\emptyset$ and show that it is a labelled bisimulation. Let $\mathcal{N}$ be the set $\{n_1, \ldots, n_m\}$, then $s_1 \mathcal{R}_\mathcal{N} s_2$ if there exist $\mathrm{m}_1, \ldots, \mathrm{m}_m$ fresh such that

$$[n_1, \ldots, n_m]\,(s_1 \mid \mathrm{m}_1!\langle n_1 \rangle \mid \ldots \mid \mathrm{m}_m!\langle n_m \rangle) \simeq_m [n_1, \ldots, n_m]\,(s_2 \mid \mathrm{m}_1!\langle n_1 \rangle \mid \ldots \mid \mathrm{m}_m!\langle n_m \rangle)$$

Take $s_1 \mathcal{R}_{\mathcal{N}} s_2$ and a transition $s_1 \xrightarrow{\alpha} s_1'$; we then reason by case analysis on α. For the sake of simplicity, we consider in detail the case where $\mathcal{N} = \emptyset$, i.e. we assume $s_1 \simeq_m s_2$. We have to consider three cases, one for each clause in Definition 4.

- $\alpha = \emptyset$.
  By computation closure of $\simeq_m$, we have that $s_2 \xrightarrow{\emptyset} s_2'$ and $s_1' \simeq_m s_2'$. $\simeq_m \subseteq \mathcal{R}_\emptyset$, hence $s_1' \mathcal{R}_\emptyset s_2'$, as required.
- $\alpha = \mathtt{n} \lhd [\bar{n}]\, \bar{v}$.
  We consider first the case where all sent values are not restricted names, i.e. $\alpha = \mathtt{n} \lhd \bar{v}$. We let the context $\mathbb{C} = [\![\cdot]\!] \mid \mathtt{n}?\bar{v}.\mathtt{m}!\langle\rangle \mid \mathtt{m}?\langle\rangle$ for $\mathtt{m}$ fresh, and consider the computation $\mathbb{C}[\![s_1]\!] \xrightarrow{\emptyset} \mathbb{C}'[\![s_1']\!]$, where $\mathbb{C}' = [\![\cdot]\!] \mid \mathtt{m}!\langle\rangle \mid \mathtt{m}?\langle\rangle$. By hypothesis, $\mathbb{C}[\![s_2]\!] \xrightarrow{\emptyset} s$ and $\mathbb{C}'[\![s_1']\!] \simeq_m s$. Since $\mathtt{m}$ is fresh, this fact implies that $s_2 \xrightarrow{\mathtt{n} \lhd \bar{v}} s_2'$ and $s \equiv \mathbb{C}'[\![s_2']\!]$, otherwise s would not be able to exhibit a barb $\downarrow_{\mathtt{m}}$ (whereas $\mathbb{C}'[\![s_1']\!]$ can). Now, we consider the computation $\mathbb{C}'[\![s_1']\!] \xrightarrow{\emptyset} s_1'$. By hypothesis and the fact that s_1' is not be able to exhibit the barb $\downarrow_{\mathtt{m}}$, $\mathbb{C}'[\![s_2']\!] \xrightarrow{\emptyset} s_2'$ and $s_1' \simeq_m s_2'$. Since by definition $\simeq_m \subseteq \mathcal{R}_\emptyset$, then we get $s_1' \mathcal{R}_\emptyset s_2'$, as required.
  Now, we consider the more general case where $\alpha = \mathtt{n} \lhd [\bar{n}]\, \bar{v}$, with $\bar{n} = \langle n_1, \ldots, n_m \rangle$ and $m \neq 0$. For the sake of presentation, suppose that $\bar{v} = (\bar{n}, \bar{v}')$. Take the context $\mathbb{C} = [\![\cdot]\!] \mid [x_1, \ldots, x_m]\, \mathtt{n}?(x_1, \ldots, x_m, \bar{v}').(\mathtt{m}!\langle x_1\rangle \mid \ldots \mid \mathtt{m}!\langle x_m\rangle)$, for $\mathtt{m}$ fresh, and consider the computation $\mathbb{C}[\![s_1]\!] \xrightarrow{\emptyset} s_3$ with $s_3 = [\bar{n}]\,(s_1' \mid \mathtt{m}!\langle n_1\rangle \mid \ldots \mid \mathtt{m}!\langle n_m\rangle)$. By hypothesis, $\mathbb{C}[\![s_2]\!] \xrightarrow{\emptyset} s_4$ and $s_3 \simeq_m s_4$. Since $s_3 \downarrow_{\mathtt{m}}$, by barb preservation, we get $s_4 \downarrow_{\mathtt{m}}$. This fact implies that s_2 has performed an invoke activity $\mathtt{n}!(\bar{v}'', \bar{v}')$ matching $\mathtt{n}?(x_1, \ldots, x_m, \bar{v}')$. If $\bar{v}''$ would contain some non-restricted names, e.g. v''', then we could define a context $\mathbb{D} = [\![\cdot]\!] \mid \mathtt{m}?\langle v'''\rangle.\mathtt{m}'!\langle\rangle$, for $\mathtt{m}'$ fresh, that can tell s_3 and s_4 apart. Indeed, $\mathbb{D}[\![s_4]\!] \xrightarrow{\emptyset} s_5$ with $s_5 \downarrow_{\mathtt{m}'}$, while for each s_6 such that $\mathbb{D}[\![s_3]\!] \xrightarrow{\emptyset} s_6$, it holds that $s_6 \not\downarrow_{\mathtt{m}'}$, that would contradict $s_5 \simeq_m s_6$ (implied by the hypothesis $s_3 \simeq_m s_4$). Thus, $\bar{v}''$ is a tuple of restricted names and, possibly by exploiting α-conversion, we get that $s_2 \xrightarrow{\mathtt{n} \lhd [\bar{n}]\, \bar{v}} s_2'$ and $s_4 = [\bar{n}]\,(s_2' \mid \mathtt{m}!\langle n_1\rangle \mid \ldots \mid \mathtt{m}!\langle n_m\rangle)$. From $s_3 \simeq_m s_4$ and by definition of $\mathcal{F}$, we conclude that $s_1' \mathcal{R}_{\bar{n}} s_2'$.
- $\alpha = \mathtt{n} \rhd [\bar{x}]\, \bar{w}$.
  For the sake of presentation, suppose that $\bar{w} = (\bar{x}, \bar{v})$. We consider the context $\mathbb{C} = [\![\cdot]\!] \mid \mathtt{n}!(\bar{v}', \bar{v})$, for $\bar{v}'$ fresh, and the computation $\mathbb{C}[\![s_1]\!] \xrightarrow{\emptyset} s_1' \cdot \{\bar{x} \mapsto \bar{v}'\}$. By computation closure, $\mathbb{C}[\![s_2]\!] \xrightarrow{\emptyset} s_3$ and $s_1' \cdot \{\bar{x} \mapsto \bar{v}'\} \simeq_m s_3$. We have two possibilities:
  - $s_2 \xrightarrow{\mathtt{n} \rhd [\bar{x}]\, \bar{w}} s_2'$ and $s_3 = s_2' \cdot \{\bar{x} \mapsto \bar{v}'\}$;
  - $s_2 \xrightarrow{\emptyset} s_2'$ and $s_3 = s_2' \mid \mathtt{n}!(\bar{v}', \bar{v})$.

  In both cases the thesis follows from the fact that, by definition of $\mathcal{F}$, $\simeq_m \subseteq \mathcal{R}_\emptyset$.

Let us now consider the more general case $\mathcal{N} \neq \emptyset$. Let $\mathcal{N}$ be the set $\{n_1, \ldots, n_k\} = \bar{n}$. As before, we have three possibilities. We show here the most interesting case: $s_1 \xrightarrow{\mathtt{n} \lhd [\bar{m}]\, \bar{v}} s_1'$ with $\mathtt{n} \notin \mathcal{N}$, $\bar{m} = \langle m_1, \ldots, m_r \rangle$ and $\bar{v} = (\bar{m}, \bar{v}')$; the other cases can be dealt with in

a similar way. Given $\mathtt{m}_1, \ldots, \mathtt{m}_k$ fresh, since $\mathtt{n} \notin \mathcal{N}$, we get that $[\bar{n}](s_1 \mid \mathtt{m}_1!\langle n_1\rangle \mid \ldots \mid \mathtt{m}_k!\langle n_k\rangle) \xrightarrow{\mathtt{n} \lhd [\bar{m}]\,\bar{v}} [\bar{n}](s_1' \mid \mathtt{m}_1!\langle n_1\rangle \mid \ldots \mid \mathtt{m}_k!\langle n_k\rangle)$. Now, consider the context $\mathbb{C} = [\![\cdot]\!] \mid [x_1, \ldots, x_r]\,\mathtt{n}?(\langle x_1, \ldots, x_r\rangle, \bar{v}').(\mathtt{m}'!\langle x_1\rangle \mid \ldots \mid \mathtt{m}'!\langle x_r\rangle)$, for $\mathtt{m}'$ fresh, and the computation $\mathbb{C}[\![[\bar{n}](s_1 \mid \mathtt{m}_1!\langle n_1\rangle \mid \ldots \mid \mathtt{m}_k!\langle n_k\rangle)]\!] \xrightarrow{\emptyset} s_3$ with $s_3 = [\bar{n}, \bar{m}](s_1' \mid \mathtt{m}_1!\langle n_1\rangle \mid \ldots \mid \mathtt{m}_k!\langle n_k\rangle \mid \mathtt{m}'!\langle m_1\rangle \mid \ldots \mid \mathtt{m}'!\langle m_r\rangle)$. As we have done for the case $\mathcal{N} = \emptyset$, we can prove that $\mathbb{C}[\![[\bar{n}](s_2 \mid \mathtt{m}_1!\langle n_1\rangle \mid \ldots \mid \mathtt{m}_k!\langle n_k\rangle)]\!] \xrightarrow{\emptyset} s_4$, $[\bar{n}](s_2 \mid \mathtt{m}_1!\langle n_1\rangle \mid \ldots \mid \mathtt{m}_k!\langle n_k\rangle) \xrightarrow{\mathtt{n} \lhd [\bar{m}]\,\bar{v}} [\bar{n}](s_2' \mid \mathtt{m}_1!\langle n_1\rangle \mid \ldots \mid \mathtt{m}_k!\langle n_k\rangle)$, $s_4 = [\bar{n}, \bar{m}](s_2' \mid \mathtt{m}_1!\langle n_1\rangle \mid \ldots \mid \mathtt{m}_k!\langle n_k\rangle \mid \mathtt{m}'!\langle m_1\rangle \mid \ldots \mid \mathtt{m}'!\langle m_r\rangle)$, and $s_3 \simeq_m s_4$. From this and by definition of $\mathcal{F}$, we conclude that $s_1' \mathcal{R}_{\bar{n} \cup \bar{m}} s_2'$. □

A.2 μCOWS

Lemma A1 (Lemma 3) *Let s_1 and s_2 be two μ*COWS *closed terms and $\mathcal{R}$ be a relation belonging to a labelled bisimulation such that $s_1 \mathcal{R} s_2$. Then,* $\mathrm{noConf}(s_1, \mathtt{n}, \bar{v}, \ell) = \mathrm{noConf}(s_2, \mathtt{n}, \bar{v}, \ell)$ *for any* $\mathtt{n}$, $\bar{v}$ *and* ℓ, *with* $\ell \leqslant |\bar{v}|$.

Proof. The proof proceeds by contradiction. Suppose that there exists $\mathtt{n}$, $\bar{v}$ and $\ell \leqslant |\bar{v}|$ such that $\mathrm{noConf}(s_1, \mathtt{n}, \bar{v}, \ell) =$ **false** and $\mathrm{noConf}(s_2, \mathtt{n}, \bar{v}, \ell) =$ **true**. By definition, $\mathrm{noConf}(s_1, \mathtt{n}, \bar{v}, \ell) =$ **false** implies that there exists a context $\mathbb{C}$ such that $s_1 = \mathbb{C}[\![\mathtt{n}?\bar{w}.s]\!]$ and s_1 can immediately perform the receive activity $\mathtt{n}?\bar{w}$ and $|\mathcal{M}(\bar{w}, \bar{v})| < \ell$. This means that there exists s_1' such that $s_1 \xrightarrow{\mathtt{n} \rhd [\bar{x}]\,\bar{w}} s_1'$ for some $\bar{x} \subseteq \bar{w}$. Since $|\mathcal{M}(\bar{w}, \bar{v})| < \ell \leqslant |\bar{v}|$, we have that $|\ \bar{x}\ | \neq |\ \bar{w}\ |$. Hence, since $s_1 \mathcal{R} s_2$ for $\mathcal{R}$ belonging to a labelled bisimulation, there exists s_2' such that $s_2 \xrightarrow{\mathtt{n} \rhd [\bar{x}]\,\bar{w}} s_2'$. From this, we get that $s_2 = \mathbb{D}[\![\mathtt{n}?\bar{w}.s]\!]$ for some context $\mathbb{D}$ such that s_2 can immediately perform the receive activity $\mathtt{n}?\bar{w}$. Thus, by definition of predicate $\mathrm{noConf}(\_, \_, \_, \_)$, we obtain $\mathrm{noConf}(s_2, \mathtt{n}, \bar{v}, \ell) =$ **false**, that is a contradiction. Of course, the case where $\mathrm{noConf}(s_1, \mathtt{n}, \bar{v}, \ell) =$ **true** and $\mathrm{noConf}(s_2, \mathtt{n}, \bar{v}, \ell) =$ **false** proceeds as before. □

Theorem A4 (Theorem 4) *$\sim_\mu$ is a congruence for μ*COWS *closed terms.*

Proof. We shall prove that, given two μCOWS closed terms s_1 and s_2, if $s_1 \sim_\mu s_2$ then $\mathbb{C}[\![s_1]\!] \sim_\mu \mathbb{C}[\![s_2]\!]$ for every (possibly open) context $\mathbb{C}$. The proof is by induction on the structure of the context $\mathbb{C}$ and proceeds similarly to that of Theorem A1. Here, we take a look only at the most relevant case of the inductive step, i.e. the case $\mathbb{C} = \mathbb{D} \mid s$ regarding the parallel composition. By induction, we may assume that $\mathbb{D}[\![s_1]\!] \sim_\mu \mathbb{D}[\![s_2]\!]$. If $\mathbb{D}$ is a closed context, then there exists a bisimulation $\mathcal{F}$ such that $\mathbb{D}[\![s_1]\!] \mathcal{R}_\emptyset \mathbb{D}[\![s_2]\!]$ with $\mathcal{R}_\emptyset \in \mathcal{F}$. By Lemma 2, we can prove the thesis by showing that the family of relations

$$\mathcal{F}' = \{\, \mathcal{R}'_{\mathcal{N}'} \ :\ \mathcal{R}_{\mathcal{N}} \in \mathcal{F}\,,\ \mathcal{N} \subseteq \mathcal{N}' \,\}$$

$$\mathcal{R}'_{\mathcal{N}'} = \{\, (\,[\bar{n}](s' \mid s), [\bar{n}](s'' \mid s)\,) \ :\ s' \mathcal{R}_{\mathcal{N}} s''\,, \text{ for some } \bar{n} \text{ and } s \text{ s.t. } \mathcal{N} \cap \mathrm{re}(s) = \emptyset \,\}$$

where $\mathrm{re}(s)$ is the set of the receiving endpoint used in s, is a bisimulation up-to $\equiv$. Let us consider a relation $\mathcal{R}'_{\mathcal{N}'} \in \mathcal{F}'$. If $[\bar{n}](s' \mid s) \xrightarrow{\alpha} s'''$, then the proof proceeds by case analysis on α. We consider here only a few relevant cases.

– $\alpha = \mathtt{n}\,\emptyset\,\ell\,\bar{v}$, $s \xrightarrow{\alpha} s''''$ and $s''' = [\bar{n}]\,(s' \mid s'''')$.
By rule *(par$_{com}$)*, since $s \xrightarrow{\mathtt{n}\,\emptyset\,\ell\,\bar{v}} s''''$ and $s' \mid s \xrightarrow{\mathtt{n}\,\emptyset\,\ell\,\bar{v}} s' \mid s''''$, we get that $\mathrm{noConf}(s', \mathtt{n}, \bar{v}, \ell) = \mathbf{true}$. Since $s'\mathcal{R}_{\mathcal{N}}s''$ for a relation $\mathcal{R}_{\mathcal{N}}$ belonging to the bisimulation $\mathcal{F}$, by Lemma A1 we have that $\mathrm{noConf}(s'', \mathtt{n}, \bar{v}, \ell) = \mathbf{true}$. Hence, by rules *(par$_{com}$)* and *(del$_2$)*, $[\bar{n}]\,(s'' \mid s) \xrightarrow{\mathtt{n}\,\emptyset\,\ell\,\bar{v}} [\bar{n}]\,(s'' \mid s'''')$. By definition of $\mathcal{F}'$, we conclude that $[\bar{n}]\,(s' \mid s'''')\,\mathcal{R}'_{\mathcal{N}'}\,[\bar{n}]\,(s'' \mid s'''')$.

– $\alpha = \mathtt{n}\,\emptyset\ |\bar{v}|\ \bar{v}$, $s' \xrightarrow{\alpha} s_3$ and $s''' = [\bar{n}]\,(s_3 \mid s)$.
By rule *(par$_{com}$)*, since $s' \xrightarrow{\mathtt{n}\,\emptyset\,|\bar{v}|\,\bar{v}} s_3$ and $s' \mid s \xrightarrow{\mathtt{n}\,\emptyset\,|\bar{v}|\,\bar{v}} s_3 \mid s$, we get that $\mathrm{noConf}(s, \mathtt{n}, \bar{v}, \ell) = \mathbf{true}$. Since $s'\mathcal{R}_{\mathcal{N}}s''$ for a relation $\mathcal{R}_{\mathcal{N}}$ belonging to the bisimulation $\mathcal{F}$, we have two possibilities:
  1. there exists s_4 such that $s'' \xrightarrow{\mathtt{n}\,\emptyset\,|\bar{v}|\,\bar{v}} s_4$ and $s_3\mathcal{R}_{\mathcal{N}}s_4$. Since $\mathrm{noConf}(s, \mathtt{n}, \bar{v}, \ell) = \mathbf{true}$ and by rules *(par$_{com}$)* and *(del$_2$)*, $[\bar{n}]\,(s'' \mid s) \xrightarrow{\mathtt{n}\,\emptyset\,|\bar{v}|\,\bar{v}} [\bar{n}]\,(s_4 \mid s)$. By definition of $\mathcal{F}'$, we conclude that $[\bar{n}]\,(s_3 \mid s)\,\mathcal{R}'_{\mathcal{N}'}\,[\bar{n}]\,(s_4 \mid s)$.
  2. there exists s_4 such that $s'' \xrightarrow{\emptyset} s_4$ and $s_3\mathcal{R}_{\mathcal{N}}s_4$. By rules *(par$_2$)* and *(del$_2$)*, $[\bar{n}]\,(s'' \mid s) \xrightarrow{\emptyset} [\bar{n}]\,(s_4 \mid s)$. By definition of $\mathcal{F}'$, we conclude that $[\bar{n}]\,(s_3 \mid s)\,\mathcal{R}'_{\mathcal{N}'}\,[\bar{n}]\,(s_4 \mid s)$.

– $\alpha = \mathtt{n}\,\emptyset\,\ell\,\bar{v}$, $\ell \neq |\bar{v}|$, $s' \xrightarrow{\alpha} s_3$ and $s''' = [\bar{n}]\,(s_3 \mid s)$.
Like the case 1 above.

By letting $s' = \mathbb{D}[\![s_1]\!]$ and $s'' = \mathbb{D}[\![s_2]\!]$, we obtain the thesis. We can generalise to the case where $\mathbb{C}$ is an open context as shown in Theorem A1. □

Theorem A5 (Soundness of $\sim_\mu$ w.r.t. $\simeq_\mu$, Theorem 5) *Given two μ*COWS *closed terms s_1 and s_2, if $s_1 \sim_\mu s_2$ then $s_1 \simeq_\mu s_2$.*

Proof. By Theorem A4, we have that $\sim_\mu$ is context closed. Thus, we only need to prove that $\sim_\mu$ is barb preserving and computation closed.

– *Barb preservation.* Suppose $s_1 \downarrow_{\mathtt{n}}$. Then, by Definition 1, this means that $s_1 \xrightarrow{\mathtt{n} \lhd [\bar{n}]\,\bar{v}} s_1'$, for some s_1', $\bar{n}$ and $\bar{v}$. Since $\sim_\mu = \sim_\mu^{\emptyset}$, by Definition 8, we get that $s_2 \xrightarrow{\mathtt{n} \lhd [\bar{n}]\,\bar{v}} s_2'$, for some s_2'. Thus, by Definition 1, $s_2 \downarrow_{\mathtt{n}}$.
– *Computation closure.* By hypothesis, there exists a labelled bisimulation $\mathcal{F}$ such that $s_1\mathcal{R}_\emptyset s_2$ with $\mathcal{R}_\emptyset \in \mathcal{F}$. Thus, if $s_1 \xrightarrow{\emptyset} s_1'$, by Definition 8, we get that $s_2 \xrightarrow{\emptyset} s_2'$ and $s_1'\mathcal{R}_\emptyset s_2'$, which means that $s_1' \sim_\mu s_2'$. Similarly, if $s_1 \xrightarrow{\mathtt{n}\,\emptyset\,\ell\,\bar{v}} s_1'$ then, by Definition 8, we get that $s_2 \xrightarrow{\alpha} s_2'$ and $s_1' \sim_\mu s_2'$, where either $\alpha = \mathtt{n}\,\emptyset\,\ell\,\bar{v}$ or $(\ell = |\bar{v}| \wedge \alpha = \emptyset)$.

□

Theorem A6 (Completeness of $\sim_\mu$ w.r.t. $\simeq_\mu$, Theorem 6) *Given two μ*COWS *closed terms s_1 and s_2, if $s_1 \simeq_\mu s_2$ then $s_1 \sim_\mu s_2$.*

Proof. We define a family of relations $\mathcal{F} = \{\mathcal{R}_{\mathcal{N}} \ : \ \mathcal{N} \text{ set of names}\}$ such that $\simeq_\mu$ is included in $\mathcal{R}_\emptyset$, and show that it is a labelled bisimulation. Let $\mathcal{N}$ be the set $\{n_1, \ldots, n_m\}$, then $s_1 \mathcal{R}_{\mathcal{N}} s_2$ if there exist $\mathtt{m}_1, \ldots, \mathtt{m}_m$ fresh such that

$$[n_1, \ldots, n_m]\,(s_1 \mid \mathtt{m}_1!\langle n_1\rangle \mid \ldots \mid \mathtt{m}_m!\langle n_m\rangle) \simeq_\mu [n_1, \ldots, n_m]\,(s_2 \mid \mathtt{m}_1!\langle n_1\rangle \mid \ldots \mid \mathtt{m}_m!\langle n_m\rangle)$$

Take $s_1 \mathcal{R}_{\mathcal{N}} s_2$ and a transition $s_1 \xrightarrow{\alpha} s_1'$; we then reason by case analysis on α. For the sake of simplicity, we consider here the case where $\mathcal{N} = \emptyset$ (the general case can be dealt with in a similar way, as shown in the proof of Theorem 3), i.e. we assume $s_1 \simeq_\mu s_2$. We have to consider the following possibilities.

- $\alpha = \emptyset$.
  By computation closure of $\simeq_\mu$, we have that $s_2 \xrightarrow{\emptyset} s_2'$ and $s_1' \simeq_\mu s_2'$. $\simeq_\mu \subseteq \mathcal{R}_\emptyset$, hence $s_1' \mathcal{R}_\emptyset s_2'$, as required.
- $\alpha = \mathtt{n}\,\emptyset\,\ell\,\bar{v}$ and $\ell \neq |\bar{v}|$.
  By computation closure of $\simeq_\mu$, we have that $s_2 \xrightarrow{\mathtt{n}\,\emptyset\,\ell\,\bar{v}} s_2'$ and $s_1' \simeq_\mu s_2'$. As before, we obtain $s_1' \mathcal{R}_\emptyset s_2'$.
- $\alpha = \mathtt{n}\,\emptyset\ |\bar{v}|\ \bar{v}$.
  By computation closure of $\simeq_\mu$, we have two possibilities:
  1. $s_2 \xrightarrow{\mathtt{n}\,\emptyset\,\sharp\,\bar{v}} s_2'$ and $s_1' \simeq_\mu s_2'$; hence, $s_1' \mathcal{R}_\emptyset s_2'$.
  2. $s_2 \xrightarrow{\emptyset} s_2'$ and $s_1' \simeq_\mu s_2'$; hence, $s_1' \mathcal{R}_\emptyset s_2'$.
- $\alpha = \mathtt{n} \lhd [\bar{n}]\,\bar{v}$.
  We consider first the case where all sent values are not restricted names, i.e. $\alpha = \mathtt{n} \lhd \bar{v}$. We let the context $\mathbb{C} = [\![\cdot]\!] \mid \mathtt{n}?\bar{v}.\,\mathtt{m}!\langle\rangle \mid \mathtt{m}?\langle\rangle$ for $\mathtt{m}$ fresh, and consider the computation $\mathbb{C}[\![s_1]\!] \xrightarrow{\emptyset} \mathbb{C}'[\![s_1']\!]$, where $\mathbb{C}' = [\![\cdot]\!] \mid \mathtt{m}!\langle\rangle \mid \mathtt{m}?\langle\rangle$. By hypothesis, $\mathbb{C}[\![s_2]\!] \xrightarrow{\emptyset} s$ and $\mathbb{C}'[\![s_1']\!] \simeq_\mu s$. This fact implies that $s_2 \xrightarrow{\mathtt{n} \lhd \bar{v}} s_2'$ and $s \equiv \mathbb{C}'[\![s_2']\!]$, otherwise s would not be able to exhibit a barb $\downarrow_{\mathtt{m}}$ (whereas $\mathbb{C}'[\![s_1']\!]$ can). Now, we consider the the computation $\mathbb{C}'[\![s_1']\!] \xrightarrow{\emptyset} s_1'$. By hypothesis and the fact that s_1' is not be able to exhibit the barb $\downarrow_{\mathtt{m}}$, $\mathbb{C}'[\![s_2']\!] \xrightarrow{\emptyset} s_2'$ and $s_1' \simeq_\mu s_2'$. Since by definition $\simeq_\mu \subseteq \mathcal{R}_\emptyset$, then we get $s_1' \mathcal{R}_\emptyset s_2'$, as required.
  Now, we consider the more general case where $\alpha = \mathtt{n} \lhd [\bar{n}]\,\bar{v}$, with $\bar{n} = \langle n_1, \ldots, n_m\rangle$ and $m \neq 0$. For the sake of presentation, suppose that $\bar{v} = (\bar{n}, \bar{v}')$. Take the context $\mathbb{C} = [\![\cdot]\!] \mid [x_1, \ldots, x_m]\,\mathtt{n}?(x_1, \ldots, x_m, \bar{v}').\,(\mathtt{m}!\langle x_1\rangle \mid \ldots \mid \mathtt{m}!\langle x_m\rangle)$, for $\mathtt{m}$ fresh, and the computation $\mathbb{C}[\![s_1]\!] \xrightarrow{\mathtt{n}\,\emptyset\,m\,\bar{v}} s_3$ with $s_3 = [\bar{n}]\,(s_1' \mid \mathtt{m}!\langle n_1\rangle \mid \ldots \mid \mathtt{m}!\langle n_m\rangle)$. Since $|\bar{v}| > m$, by hypothesis, $\mathbb{C}[\![s_2]\!] \xrightarrow{\mathtt{n}\,\emptyset\,m\,\bar{v}} s_4$ and $s_3 \simeq_\mu s_4$. Since $s_3 \downarrow_{\mathtt{m}}$, by barb preservation, we get $s_4 \downarrow_{\mathtt{m}}$. This fact implies that s_2 has performed an invoke activity $\mathtt{n}!(\bar{v}'', \bar{v}')$ matching $\mathtt{n}?(x_1, \ldots, x_m, \bar{v}')$. If $\bar{v}''$ would contain some non-restricted names, e.g. v''', then we could define a context $\mathbb{D} = [\![\cdot]\!] \mid \mathtt{m}?\langle v'''\rangle.\,\mathtt{m}'!\langle\rangle$, for $\mathtt{m}'$ fresh, that can tell s_3 and s_4 apart. Indeed, $\mathbb{D}[\![s_4]\!] \xrightarrow{\emptyset} s_5$ with $s_5 \downarrow_{\mathtt{m}'}$, while for each s_6 such that $\mathbb{D}[\![s_3]\!] \xrightarrow{\emptyset} s_6$, it holds that $s_6 \not\downarrow_{\mathtt{m}'!}$, that would contradict $s_5 \simeq_\mu s_6$ (implied by the hypothesis $s_3 \simeq_\mu s_4$). Thus, $\bar{v}''$ is a tuple of restricted names and, possibly by exploiting α-conversion, we get that $s_2 \xrightarrow{\mathtt{n} \lhd [\bar{n}]\,\bar{v}} s_2'$ and $s_4 = [n]\,(s_2' \mid \mathtt{m}!\langle n_1\rangle \mid \ldots \mid \mathtt{m}!\langle n_m\rangle)$. From $s_3 \simeq_\mu s_4$ and by definition of $\mathcal{F}$, we conclude that $s_1' \mathcal{R}_{\bar{n}} s_2'$.

– $\alpha = \mathtt{n} \triangleright [\bar{x}]\,\bar{w}$.
We have the following possibilities:
- $\alpha = \mathtt{n} \triangleright \langle\rangle$.
We consider the context $\mathbb{C} = [\![\cdot]\!] \mid \mathtt{n}!\langle\rangle$ and the computation $\mathbb{C}[\![s_1]\!] \xrightarrow{\emptyset} s_1'$. By hypothesis, $\mathbb{C}[\![s_2]\!] \xrightarrow{\emptyset} s$ and $s_1' \simeq_\mu s$. We have two possibilities:
  1. $s_2 \xrightarrow{\mathtt{n} \triangleright \langle\rangle} s_2'$ and $s = s_2'$;
  2. $s_2 \xrightarrow{\emptyset} s_2'$ and $s = s_2' \mid \mathtt{n}!\langle\rangle$;

  In both cases the thesis follows from the fact that, by definition of $\mathcal{F}$, $\simeq_\mu \subseteq \mathcal{R}_\emptyset$.
- $\alpha = \mathtt{n} \triangleright [\bar{x}]\,\bar{x}$.
We consider the context $\mathbb{C} = [\![\cdot]\!] \mid \mathtt{n}!\bar{v}$ for some $\bar{v}$ such that $\mathcal{M}(\bar{x}, \bar{v}) = \sigma$, $\mathrm{noConf}(s_2, \mathtt{n}, \bar{v}, |\bar{x}|) = \mathbf{true}$ and $\mathbb{C}[\![s_1]\!] \xrightarrow{\mathtt{n}\emptyset|\bar{v}|\bar{v}} s_1' \cdot \sigma$. By hypothesis, we have two possibilities:
  1. $\mathbb{C}[\![s_2]\!] \xrightarrow{\mathtt{n}\emptyset|\bar{v}|\bar{v}} s$ and $s_1 \cdot \sigma \simeq_\mu s$. There are two cases:
     (a) $s_2 \xrightarrow{\mathtt{n} \triangleright [\bar{x}]\,\bar{x}} s_2'$ and $s = s_2' \cdot \sigma$;
     (b) $s_2 \xrightarrow{\mathtt{n}\emptyset|\bar{v}|\bar{v}} s_2''$. Since the length of the generated substitution is $|\bar{v}|$ (i.e. the argument of the executed receive is a tuple of only variables), by rule *(com)*, $s_2 \xrightarrow{\mathtt{n} \triangleright [\bar{x}]\,\bar{x}} s_2'$ and, hence, we can proceed as in the previous case.
  2. $\mathbb{C}[\![s_2]\!] \xrightarrow{\emptyset} s$ and $s_1' \simeq_\mu s$. This means that $s_2 \xrightarrow{\emptyset} s_2'$ and $s = s_2' \mid \mathtt{n}!\bar{v}$.

  In both cases the thesis follows from the fact that, by definition of $\mathcal{F}$, $\simeq_\mu \subseteq \mathcal{R}_\emptyset$.
- $\alpha = \mathtt{n} \triangleright [\bar{x}]\,\bar{w}$ with $|\bar{x}| \neq |\bar{w}|$.
Without loss of generality, we may assume that $\bar{w} = (\bar{x}, \bar{v})$. We consider the context $\mathbb{C} = [\![\cdot]\!] \mid \mathtt{n}!(\bar{v}', \bar{v})$ with $\bar{v}'$ such that for any receive activity $\mathtt{n}?\bar{w}'$ that s_1 and s_2 can immediately perform, with $\bar{w}'$ more or equal defined than $\bar{w}$, $\mathcal{M}(\bar{w}', (\bar{v}', \bar{v}))$ does not hold. Then, consider the transition $\mathbb{C}[\![s_1]\!] \xrightarrow{\mathtt{n}\emptyset|\bar{x}|(\bar{v}',\bar{v})} s_1' \cdot \{\bar{x} \mapsto \bar{v}'\}$. By hypothesis, $\mathbb{C}[\![s_2]\!] \xrightarrow{\mathtt{n}\emptyset|\bar{x}|(\bar{v}',\bar{v})} s$ and $s_1' \cdot \{\bar{x} \mapsto \bar{v}'\} \simeq_\mu s$. We have the following possibilities:
  1. $s_2 \xrightarrow{\mathtt{n} \triangleright [\bar{v}]\,\bar{w}'} s_2'$.
     Since the substitution generated by the communication is $|\bar{x}|$ long, we have that $|\bar{x}| = |\bar{y}|$ and, by possibly exploiting α-conversion, we get that $s_2 \xrightarrow{\mathtt{n} \triangleright [\bar{x}]\,\bar{w}''} s_2'$ with $\bar{w}''$ obtained from $\bar{w}'$ by replacing $\bar{y}$ with $\bar{x}$. From this, we have that $\bar{w}''$ is as defined as $\bar{w}$. Moreover, by rule *(com)*, we have that $\mathcal{M}(\bar{w}'', (\bar{v}', \bar{v}))$ holds. Hence, by the constraints on $\bar{v}'$, we get $\bar{w}'' = \bar{w}$. Finally, $s_2 \xrightarrow{\mathtt{n} \triangleright [\bar{x}]\,\bar{w}} s_2'$ and $s = s_2' \cdot \{\bar{x} \mapsto \bar{v}'\}$.
  2. $s_2 \xrightarrow{\mathtt{n}\emptyset|\bar{x}|(\bar{v}',\bar{v})} s_2'$ and $s = s_2' \mid \mathtt{n}!(\bar{v}', \bar{v})$.
     By rule *(com)* and by following the same reasoning used before, we can state that $s_2 \xrightarrow{\mathtt{n} \triangleright [\bar{x}]\,\bar{w}} s_2''$. Therefore, $\mathbb{C}[\![s_2]\!] \xrightarrow{\mathtt{n}\emptyset|\bar{x}|(\bar{v}',\bar{v})} s_2'' \cdot \{\bar{x} \mapsto \bar{v}'\}$ and $s_1' \cdot \{\bar{x} \mapsto \bar{v}'\} \simeq_\mu s_2'' \cdot \{\bar{x} \mapsto \bar{v}'\}$.

  In both cases the thesis follows from the fact that, by definition of $\mathcal{F}$, $\simeq_\mu \subseteq \mathcal{R}_\emptyset$.

□

A.3 COWS

Theorem A7 (Theorem 7) *$\sim$ is a congruence for* COWS.

Proof. We shall prove that, given two COWS closed terms s_1 and s_2, if $s_1 \sim s_2$ then $\mathbb{C}[\![s_1]\!] \sim \mathbb{C}[\![s_2]\!]$ for every context $\mathbb{C}$. The proof is by induction on the structure of the context $\mathbb{C}$. The base case, i.e. whenever $\mathbb{C} = [\![\cdot]\!]$, is trivial. For the inductive case, we take a look at two cases:

- $\mathbb{C} = [k]\,\mathbb{D}$.
  If $\mathbb{D}$ is a closed context, then $[k]\,\mathbb{D} \equiv \mathbb{D}$ and, by induction, we immediately conclude. Instead, if $\mathbb{D}$ contains the free killer label k at most[3], by induction we may assume that $\mathbb{D}[\![s_1]\!] \sim \mathbb{D}[\![s_2]\!]$. This means that, there exists a bisimulation $\mathcal{F}$ such that $\mathbb{D}[\![s_1]\!]\mathcal{R}_\emptyset\mathbb{D}[\![s_2]\!]$ for some $\mathcal{R}_\emptyset \in \mathcal{F}$. Now, consider the following family of relations:

  $$\{\ \{\,(\,[k]\,s, [k]\,s'\,)\ :\ s\,\mathcal{R}_\mathcal{N}s'\ ,\ s \text{ and } s' \text{satisfy } (*)\,\} \cup \mathcal{R}_\mathcal{N}\ \ :\ \mathcal{R}_\mathcal{N} \in \mathcal{F}\ \}$$

  where property $(*)$ states that if $s \xrightarrow{k} s''$ then $s' \xrightarrow{k} s'''$ and $s''\mathcal{R}_\mathcal{N}s'''$. The proof proceed by proving that the above family of relations is a bisimulation up-to $\equiv$. Indeed, by letting $s = \mathbb{D}[\![s_1]\!]$ and $s' = \mathbb{D}[\![s_2]\!]$, we obtain the thesis. In fact, since s_1 and s_2 are closed terms, $\mathbb{D}[\![s_1]\!]$ and $\mathbb{D}[\![s_2]\!]$ satisfy property $(*)$.

  Thus, let us consider a relation $\mathcal{R}'_\mathcal{N}$ belonging to the above family. If $[k]\,s \xrightarrow{\alpha} s_3$, we proceed by case analysis on α. The most interesting case is $\alpha = \dagger$, with $s \xrightarrow{k} s_4$ and $s_3 = [k]\,s_4$. By property (*), $s' \xrightarrow{k} s_5$ and $s_4\mathcal{R}_\mathcal{N}s_5$. From this, we get that $[k]\,s' \xrightarrow{\dagger} [k]\,s_5$ and, by definition of $\mathcal{R}'_\mathcal{N}$, we conclude $[k]\,s_4\mathcal{R}'_\mathcal{N}[k]\,s_5$. The remaining cases can be easily proved by exploiting the fact that $s\,\mathcal{R}_\mathcal{N}s'$.
- $\mathbb{C} = \{\!|\mathbb{D}|\!\}$.
  By induction, we may assume that $\mathbb{D}[\![s_1]\!] \sim \mathbb{D}[\![s_2]\!]$. If $\mathbb{C}$ is a closed context, then there exists a bisimulation $\mathcal{F}$ such that $\mathbb{D}[\![s_1]\!]\mathcal{R}_\emptyset\mathbb{D}[\![s_2]\!]$ for some $\mathcal{R}_\emptyset \in \mathcal{F}$. Thus, we can prove the thesis by showing that $\{\,(\{\!|s|\!\}, \{\!|s'|\!\})\ :\ s\mathcal{R}_\mathcal{N}s'\,\}$, $\mathcal{R}_\mathcal{N} \in \mathcal{F}\,\}$ is a bisimulation up-to $\equiv$. Instead, if $\mathbb{D}$ contains free variables, we must consider $\mathbb{D}[\![s_1]\!]$ and $\mathbb{D}[\![s_2]\!]$ under all possible substitution for such variables. □

[3] The case where $\mathbb{D}$ contains more than one free killer label can be dealt with in a similar way, while the case where $\mathbb{D}$ contains free variables can be dealt with as in Theoerm A1.