A Process Calculus for Mobile Ad Hoc Networks[☆]

Anu Singh, C. R. Ramakrishnan, Scott A. Smolka

Department of Computer Science
Stony Brook University
Stony Brook, NY 11794-4400, USA

Abstract

We present the ω-calculus, a process calculus for formally modeling and reasoning about *Mobile Ad Hoc Wireless Networks* (MANETs) and their protocols. The ω-calculus naturally captures essential characteristics of MANETs, including the ability of a MANET node to broadcast a message to any other node within its physical transmission range (and no others), and to move in and out of the transmission range of other nodes in the network. A key feature of the ω-calculus is the separation of a node's communication and computational behavior, described by an ω-process, from the description of its physical transmission range, referred to as an ω-process *interface*.

Our main technical results are as follows. We give a formal operational semantics of the ω-calculus in terms of labeled transition systems and show that the state reachability problem is decidable for finite-control ω-processes. We also prove that the ω-calculus is a conservative extension of the π-calculus, and that late bisimulation equivalence (appropriately lifted from the π-calculus to the ω-calculus) is a congruence. Congruence results are also established for a weak version of late bisimulation equivalence, which abstracts away from two types of internal actions: τ-actions, as in the π-calculus, and μ-actions, signaling node movement. We additionally define a symbolic semantics for the ω-calculus extended with the *mismatch* operator, along with a corresponding notion of symbolic bisimulation equivalence, and establish congruence results for this extension as well. Finally, we illustrate the practical utility of the calculus by developing and analyzing formal models of a leader-election protocol for MANETs and the AODV routing protocol.

Key words: Mobile ad hoc networks, Process calculi, Bisimulation, Congruence

[☆] A preliminary version of this paper appeared in COORDINATION'08.

Email addresses: anusingh@cs.sunysb.edu (Anu Singh), cram@cs.sunysb.edu (C. R. Ramakrishnan), sas@cs.sunysb.edu (Scott A. Smolka)

1. Introduction

A Mobile Ad Hoc Network (MANET) is a network of autonomous mobile nodes connected by wireless links. Each node N has a physical transmission range within which it can directly transmit data to other nodes. Any node that falls within N's transmission range is considered a *neighbor* of N. Nodes can move freely in a MANET, leading to rapid changes in the network's communication topology.

Two aspects of MANETs make them especially difficult to model using existing formal specification languages such as process algebras. First, MANETs use wireless links for local broadcast communication: a MANET node can transmit a message simultaneously to all nodes within its transmission range, but the message cannot be received by any node outside that range. Secondly, a node's neighborhood can change unpredictably due to node movement, thereby altering the set of nodes that can receive a transmitted message.

Ideally, the specification of a node's control behavior should be independent of its neighborhood information. Since, however, the eventual recipients of a local broadcast message depend on this information, a model of a MANET-based protocol given in a traditional process calculus must intermix the computation of neighborhood information with the protocol's control behavior. This tends to render such models unnatural and unnecessarily complex.

In this paper, we present the ω-calculus, a conservative extension of the π-calculus that has been designed expressly to address the MANET modeling problems outlined above. A key feature of the ω-calculus is the separation of a node's communication and computational behavior, described by an ω-process, from the description of its physical transmission range, referred to as an ω-process *interface*. This separation allows one to model the control behavior of a MANET protocol using ω-processes independently from the protocol's underlying communication topology, which is modeled using process interfaces. (A similar separation of concerns has been achieved in several recently introduced process calculi for wireless and mobile networks [13, 10, 9, 6], but not, as we argue in Section 8, as simply and naturally as in the ω-calculus.)

As discussed further in Section 2, ω-process interfaces are comprised of *groups*, which operationally function as local broadcast ports. Mobility is captured in the ω-calculus via the dynamic creation of new groups and dynamically changing process interfaces. The group-based abstraction for local broadcast in a wireless network is a natural one; it appears also in [7], where it is shown how to model MANETs in the UPPAAL model checker for timed automata.

Main Contributions. The rest of the paper is organized around our main technical results, which include the following:

- Section 2 provides an informal introduction to the basic features of the ω-calculus.
- Section 3 presents the formal operational semantics of the ω-calculus in terms of labeled transition systems and structural-congruence rules. The calculus is presented in three stages: ω_0, the core version of the calculus, focuses on local broadcast and mobility; ω_1 extends ω_0 with unicast communication

Figure 1: Multiple views of a MANET network.

and scope extrusion; ω_2 extends ω_1 by allowing multi-threaded behavior at the process level. We shall henceforth use the term "ω-calculus" to refer to ω_2, the most general version of the calculus. We in fact show in Section 4 that ω_2 is a conservative extension of the π-calculus.

- Section 4 defines bisimulation equivalence for the ω-calculus and proves that it is a congruence. We obtain similar results for a weak version of bisimulation, which treats as unobservable two types of internal actions: τ-actions, as in the π-calculus, and μ-actions, signaling node movement.
- Section 5 extends the transitional semantics of the ω-calculus to a symbolic one in the presence of a mismatch operator. Symbolic bisimulation equivalence is also defined and is shown to be a congruence.
- Sections 6 presents our Prolog encoding of the transitional semantics of the ω-calculus.
- Section 7 illustrates the practical utility of the calculus by developing and analyzing formal ω-calculus models for two algorithms for MANETs, namely a leader-election algorithm [20] and the AODV routing protocol [16].

Comment (2a) Done. Throughout the paper: "Bisimulation is not a congruence in general, bisimilarity, i.e. the largest bisimulation instead is"

Comment (2b) Done. Line too long

Section 8 considers related work and Section 9 offers our concluding remarks.

2. The ω-Calculus: An Informal Introduction

As an illustrative example of the ω-calculus, consider the MANET of Fig. 1(a) comprising the four nodes N_1, N_2, N_3, N_4. The dotted circle centered around a node indicates the node's transmission range. Thus, N_1 is within the transmission range of N_2, N_3, and N_4 and vice versa, and N_2 and N_4 are in each other's transmission range. We assume that the transmission ranges of all nodes are identical, and hence connectivity is symmetric. The assumption of symmetry makes the notation cleaner, although the assumption can be readily removed, as discussed later in this section. Fig. 1(b) highlights the *maximal sets of neighboring nodes* in the network, one covering N_1, N_2, and N_4, and the other covering N_1 and N_3. A maximal set of neighboring nodes corresponds to a *maximal clique* in the network's node connectivity graph (Fig. 1(c)), and, equivalently, to an ω-calculus *group* (local broadcast port), as illustrated in Fig. 1(d). The set of groups to which a node is connected is specified by the *interface* of the underlying process; i.e. the process executing at the node. Thus, the ω-calculus expression for the network is the parallel composition $N_1|N_2|N_3|N_4$, where $N_1 = P_1 : \{g_1, g_2\}$, $N_2 = P_2 : \{g_1\}$, $N_3 = P_3 : \{g_2\}$, $N_4 = P_4 : \{g_1\}$, for process expressions P_1, P_2, P_3 and P_4.

Comment (2c) Done. Connectivity is assumed to be symmetric. State/discuss this assumption

Note that process interfaces may contain groups that do not correspond to maximal cliques. Groups that do not represent any additional connectivity information are redundant. Group g_2 of Fig. 2 is an example of a redundant group. A *canonical* form for ω-calculus expressions can be defined in which redundant groups are elided.

Comment (1a) Done. About removing redundant groups

Fig. 1 provides multiple views of the topology of the MANET at a particular

Figure 2: (a) Node Connectivity Graph after N_3's movement and (b) View in ω-calculus.

moment in time. As discussed below, the network topology may change over time due to node movement, a feature of MANETs captured operationally in the ω-calculus via dynamic updates of process interfaces.

Local Broadcast in the ω-calculus. The ω-calculus action to locally broadcast a value x is $\overline{\mathbf{b}}x$, while $\mathbf{r}(y)$ is the action for receiving a value y. Thus, when a process transmits a message, only the message x to be sent is included in the specification. The set of possible recipients depends on the process's current interface: only those processes that share a common group with the sender can receive the message and this information is not part of the syntax of local broadcast actions. In the example of Fig. 1, if P_2 can broadcast a message and P_1, P_3, P_4 are willing to receive it, then the expression

$$N = \mathbf{r}(x).P_1' : \{g_1, g_2\} \mid \overline{\mathbf{b}}u.P_2' : \{g_1\} \mid \mathbf{r}(y).P_3' : \{g_2\} \mid \mathbf{r}(z).P_4' : \{g_1\}$$

may evolve to

$$N = P_1'\{u/x\} : \{g_1, g_2\} \mid P_2' : \{g_1\} \mid \mathbf{r}(y).P_3' : \{g_2\} \mid P_4'\{u/z\} : \{g_1\}$$

Observe that P_3 does not receive the message since N_3 is not in N_2's neighborhood. It should be noted that communication is assumed to be lossy, and hence even nodes that are within a sender's transmission range may not receive a message.

Comment (2d) Done. Mention that even nodes in the trasmission range may not receive messages.

When the interfaces of two nodes share a group name, the nodes are in each others' transmission ranges. We can remove the assumption of symmetric connections by partitioning the interface into transmission and reception parts. Then a node N_1 can send a message that can be received by node N_2 if the transmission interface of N_1 overlaps with the reception interface of N_2. Note that N_2's transmission interface and N_1's reception interface may be disjoint. This captures the scenario where N_2 is in N_1's transmission range, but N_1 is not in N_2's transmission range. While assymmetric connections can be handled in principle, this introduces notational clutter. Consequently, our technical development in this paper assumes symmetric connections.

Node mobility in the ω-calculus. Node mobility is captured through the dynamic creation of new groups and dynamically changing process interfaces. Fig. 2 shows the topology of the network of Fig. 1 after N_3 moves away from N_1's transmission range and into N_4's transmission range. N_3's movement means that the ω-calculus expression

$$(\nu g_1)(\nu g_2)(P_1 : \{g_1, g_2\} \mid P_2 : \{g_1\} \mid P_3 : \{g_2\} \mid P_4 : \{g_1\})$$

evolves to

$$(\nu g_1)(\nu g_2)(P_1 : \{g_1, g_2\} \mid P_2 : \{g_1\} \mid (\nu g_3)(P_3 : \{g_3\} \mid P_4 : \{g_1, g_3\}))$$

The new group g_3 in the above expression represents the new maximal set of neighboring nodes N_3 and N_4 that arises post-movement. We use the familiar νg notation for group-name scoping.

Added about mobility invariance; revised by CR

When process interfaces are allowed to change arbitrarily, the network topology may change without any restriction. Correctness properties of many MANET algorithms and protocols may hold only in certain restricted class of topologies. We equip the ω calculus to restrict node movement by imposing an invariant over a network's topology, called the connectivity invariant, which must be preserved whenever the topology changes. Note that a connectivity invariant of "true" will allow arbitrary node movement.

Nodes vs. Processes. In an ω-calculus specification, nodes typically represent physical devices; as such, the calculus does not provide a primitive for node creation. Process creation, however, is supported, as processes model programs and other executables that execute within the confines of a device.

3. Syntax and Transitional Semantics of the ω-Calculus

We begin this section by presenting the syntax and semantics of ω_0, our core calculus for MANETs. We then introduce the extensions to ω_0 that result in the more expressive ω_1- and ω_2-calculi.

3.1. Syntax of ω_0

A system description in the ω_0-calculus comprises a set of *nodes*, each of which runs a sequential *process* annotated by its *interface*. We use $\mathbf{N}$ and $\mathbf{P}$ to denote the sets of all nodes and all processes, respectively, with M, N ranging over nodes and P, Q ranging over processes. We also use names drawn from two disjoint sets: $\mathbf{Pn}$ and $\mathbf{Gn}$. The names in $\mathbf{Pn}$, called *pnames* for *process names*, are used for data values. The names in $\mathbf{Gn}$, called *gnames* for *group names*, are used for process interfaces. We use x, y, z to range over $\mathbf{Pn}$ and g (possibly subscripted) to range over $\mathbf{Gn}$. The ω_0-calculus has a two-level syntax describing nodes and processes, respectively.

The syntax of ω_0-calculus processes is defined by the following grammar:

$$\begin{array}{rcl} P & ::= & \mathit{nil} \mid \mathit{Act}.P \mid P + P \mid [x = y]P \mid A(\vec{x}) \\ \mathit{Act} & ::= & \overline{\mathbf{b}}x \mid \mathbf{r}(x) \mid \tau \end{array}$$

Action $\overline{\mathbf{b}}x$ represents the local broadcast of a value x, while the reception of a locally broadcasted value is denoted by $\mathbf{r}(x)$. Internal (silent) actions are denoted by τ. Process *nil* is the deadlocked process; $\mathit{Act}.P$ is the process that can perform action Act and then behave as P; and $+$ is the operator for nondeterministic choice. Process $[x = y]P$ (where x and y are pnames) behaves as P if names x and y match, and as *nil* otherwise. $A(\vec{x})$ denotes *process invocation*, where A is a process identifier (having a corresponding definition) and $\vec{x}$ is a comma-separated list of actual parameters (pnames) of the invocation. A process definition is of the form $A(\vec{x}) \stackrel{\text{def}}{=} P$, and associates a process identifier A and a list of formal parameters $\vec{x}$ (i.e. distinct pnames) with process expression P. Process definitions may be recursive.

The following grammar defines the syntax of ω_0-calculus node expressions:

$$M \quad ::= \quad \mathbf{0} \;\mid\; P{:}G \;\mid\; (\nu g)M \;\mid\; M|M$$

$\mathbf{0}$ is the inactive node, while $P:G$, where $G \subseteq \mathbf{Gn}$, is a node with process P having interface G. The operator (νg) is used to restrict the scopes of gnames. $M|N$ represents the parallel composition of node expressions M and N. Node expressions of the form $P:G$ are called *basic node expressions*, while those containing the restriction or parallel operator are called *structured node expressions*. Note that gnames occur only at the node level, capturing the intuition that, in an ad hoc network, the behavioral specification of a (basic) node (represented by its process) is independent of its underlying interface.

Added $G \subseteq \mathbf{Gn}$.

Free and Bound Names. For a process expression P, the set of free names and bound names of P, denoted as *fn(P)* and *bn(P)*, respectively, are defined as follows:

$fn(nil)$	$=$	$\emptyset$	$bn(nil)$	$=$	$\emptyset$
$fn(\overline{\mathbf{b}}x.P)$	$=$	$fn(P) \cup \{x\}$	$bn(\overline{\mathbf{b}}x.P)$	$=$	$bn(P)$
$fn(\mathbf{r}(x).P)$	$=$	$fn(P) \setminus \{x\}$	$bn(\mathbf{r}(x).P)$	$=$	$bn(P) \cup \{x\}$
$fn(\tau.P)$	$=$	$fn(P)$	$bn(\tau.P)$	$=$	$bn(P)$
$fn(P+Q)$	$=$	$fn(P) \cup fn(Q)$	$bn(P+Q)$	$=$	$bn(P) \cup bn(Q)$
$fn([x=y]P)$	$=$	$fn(P) \cup \{x,y\}$	$bn([x=y]P)$	$=$	$bn(P)$
$fn(A(x_1,\ldots,x_n))$	$=$	$\{x_1,\ldots,x_n\}$	$bn(A(x_1,\ldots,x_n))$	$=$	$\emptyset$

In a process definition of the form $A(\vec{x}) \stackrel{\text{def}}{=} P$, $\vec{x}$ are the only names that may occur free in P. The set of all names in a process expression P is given by $n(P)$, where $n(P) = fn(P) \cup bn(P)$. Similarly, the set of all pnames and gnames in a node expression M are denoted by $pn(M)$ and $gn(M)$, and those that occur free are denoted by $fpn(M)$ and $fgn(M)$, respectively. Gname g is bound in $(\nu g)M$, and all gnames in G are free in $P:G$. The set of all free names in a node expression M is given by $fn(M) = fpn(M) \cup fgn(M)$. An expression without free names is called *closed*. An expression that is not *closed* is said to be *open*. The theory developed in the following sections is applicable to both *open* and *closed* systems (expressions).

Comment (2f) Done. "can G be empty? If yes, then strange labels are produced when it broadcasts something, and the only if part of Th. 1 (conservative extension of the pi-calculus) is false since $P : \{g\} \xrightarrow{\mu} P : \{\}$ can be derived using congruence (with N1) and MOBILITY"

Comment (2g) Done. "please add that names in x are free in A(x)".

3.2. Transitional Semantics of ω_0

The transitional semantics of the ω_0-calculus is defined in terms of a structural congruence relation $\equiv$ (Table 1) and a labeled transition relation $\longrightarrow \subseteq \mathbf{N} \times L \times \mathbf{N}$, where $L = \{\overline{G}x, G(x), \tau, \mu \mid G \subseteq \mathbf{Gn}, x \in \mathbf{Pn}\}$ is a set of transition labels. A labeled transition $(M, \alpha, M') \in \longrightarrow$, is also represented as $M \xrightarrow{\alpha} M'$. As such, only node expressions have transitions. When a node of the form $P:G$ broadcasts a value x, it generates a transition labeled by $\overline{G}x$. When $P:G$ receives a broadcast value x, the corresponding transition label is $G(x)$. Actions μ and τ also serve as transition labels, with μ, as explained below, indicating node movement, and τ representing internal (silent) actions.

Comment (2e) Done. " I will present P rules before N rules (and move the table where it is referenced)"

For transition label α, the sets of bound names and gnames of α are denoted $bn(\alpha)$ and $gn(\alpha)$, respectively, and defined as follows:

$bn(\overline{G}x) = \emptyset$, $bn(G(x)) = \{x\}$, $bn(\mu) = \emptyset$, $bn(\tau) = \emptyset$.

P1.	$P + Q \equiv Q + P$
P2.	$(P + Q) + R \equiv P + (Q + R)$
P3.	$P \equiv Q$, if $P \equiv_\alpha Q$
N1.	$M \equiv M \mid \mathbf{0}$
N2.	$M_1 \mid M_2 \equiv M_2 \mid M_1$
N3.	$(M_1 \mid M_2) \mid M_3 \equiv M_1 \mid (M_2 \mid M_3)$
N4.	$(\nu g)M \equiv M$, if $g \notin fgn(M)$
N5.	$(\nu g)M \mid N \equiv (\nu g)(M \mid N)$, if $g \notin fgn(N)$
N6.	$(\nu g_1)(\nu g_2)M \equiv (\nu g_2)(\nu g_1)M$
N7.	$M \equiv N$, if $M \equiv_\alpha N$
N8.	$P{:}G \equiv Q{:}G$, if $P \equiv Q$
N9.	$P{:}G \equiv (\nu g)(P{:}G \cup \{g\})$, if $g \notin G$

Table 1: Structural congruence relation for the ω_0-calculus.

$gn(\overline{G}x) = G$, $gn(G(x)) = G$, $gn(\mu) = \emptyset$, $gn(\tau) = \emptyset$.

We define a label restriction operation $\alpha \setminus G$ that makes visible only those group names in α that are not in set G as follows:

Defined $\alpha \setminus G$

$$\begin{array}{lcl} \tau \setminus G & = & \tau \\ \mu \setminus G & = & \mu \\ \overline{G_1}x \setminus G_2 & = & \overline{G_1 - G_2}\, x \\ G_1(x) \setminus G_2 & = & (\mathrm{G}_1 - G_2)(x) \end{array}$$

where we use $G_1 - G_2$ to denote the set $\{g \in G_1 \mid g \notin G_2\}$.

Added side-condition $G \neq \emptyset$ to MCAST and RECV rules. Also add this side-condition to unicast rules.

We use the standard notion of substitution for names, viz. a mapping $\sigma : \mathbf{Pn} \times \mathbf{Pn}$. We also use the standard notation for application of substitution to terms. The expression $M\{y/x\}$ denotes the node expression in which all free occurrences of x are replaced by y in M, with a change of bound names if necessary to avoid any of the new name y from becoming bound in M.

Moved the discussion of invariance condition here. Changed *mobility invariant* to *connectivity invariant*. Alternatively, *topology invariant*

Process interfaces provide an abstract specification of network topology in terms of node connectivity graphs. Formally, the *node connectivity graph* of a node expression M, denoted by $\chi(M)$, is an undirected graph (V, E) such that V, the set of vertices, are the basic nodes of M (i.e. subexpressions of M of the form $P : G$) and E, the set of edges, is defined as follows. There is an edge between two vertices $P_1{:}G_1$ and $P_2{:}G_2$ of $\chi(M)$ only if P_1 and P_2's interfaces overlap; i.e. $G_1 \cap G_2 \not\equiv \emptyset$ (assuming bound names of M are unique and distinct from its free names). The node connectivity graph for the ω_0 node expression of Fig. 1(d) is given in Fig. 1(c).

Comment (1g) Done. "You should better motivate the need for the notion of mobility invariance.".

We use the notion of *connectivity invariant*, to impose different models of node movement on the calculus. A connectivity invariant is a decidable property over undirected graphs. For example, k-connectedness, for a given k, is a candidate connectivity invariant, as is `true`, indicating no constraints on node movement. We write $I(U)$ to indicate that undirected graph U possesses prop-

Rule Name	Rule	Side Condition
MCAST	$\dfrac{}{(\overline{\mathbf{b}}x.P){:}G \xrightarrow{\overline{G}x} P{:}G}$	$G \neq \emptyset$
RECV	$\dfrac{}{(\mathbf{r}(x).P){:}G \xrightarrow{G(x)} P{:}G}$	$G \neq \emptyset$
CHOICE	$\dfrac{P{:}G \xrightarrow{\alpha} P'{:}G}{(P+Q){:}G \xrightarrow{\alpha} P'{:}G}$	
MATCH	$\dfrac{P{:}G \xrightarrow{\alpha} P'{:}G}{([x{=}x]P){:}G \xrightarrow{\alpha} P'{:}G}$	
DEF	$\dfrac{P\{\vec{y}/\vec{x}\}{:}G \xrightarrow{\alpha} P'{:}G}{A(\vec{y}){:}G \xrightarrow{\alpha} P'{:}G}$	$A(\vec{x}) \stackrel{def}{=} P$

Table 2: Transition rules for ω_0-calculus basic node expressions.

erty I. We also use $I(M)$, thus overloading I, to denote $I(\chi(M))$ which means that the connectivity graph of node expression M satisifies connectivity invariant I.

Added that we define late semantics.

The transitional semantics of the ω_0-calculus is given by the inference rules of Tables 2 and 3, with the former supplying the inference rules for basic node expressions and the latter for structured node expressions. Rules CHOICE, MATCH, and DEF of Table 2 are standard. Rules MCAST and RECV of Table 2, together with COM of Table 3, define a notion of *local broadcast* communication. RECV states that a basic node with process interface G can receive a local broadcast on any gname in G. This, together with COM, means that a local-broadcast sender can synchronize with any local-broadcast receiver with whom it shares a gname (i.e. the receiver is in the transmission range of the sender). Note that a node with an empty in interface cannot perform send or receive actions. Note also that the above definition corresponds to *late* semantics due to the late instantiation of received names.

Added that send/receive actions are not permitted on empty interface.

Local-broadcast synchronization results in a local-broadcast transition label of the form $\overline{G}x$, thereby enabling other receivers to synchronize with the original send action. PAR rule indicates the interleaving semantics for actions of nodes in parallel. The first side condition is standard and is used to avoid name capture. The second side condition permits only those node movements that preserve a connectivity invariant I in a larger network context.

Added about PAR(I) rule and modified text for GNAMES-RES rules.

GNAME-RES1 and GNAME-RES2 define the effect of closing the scope of a gname. GNAME-RES1 states that a restricted gname cannot occur in a transition label. GNAME-RES2 states that when all gnames of a local-broadcast-send action are restricted, it becomes a τ-action. MCAST, GNAME-RES1

Comment (1c), (2h) for GNAME-RES1 (Done). "Shouldn't there be a side condition stating $gn(\alpha) \neq \{g\}$". "Adding the condition $gn(\alpha) \neq \{g\}$ will simplify the definition of \".

Rule Name	Rule	Side Condition
STRUCT	$\dfrac{N \equiv M \quad M \xrightarrow{\alpha} M' \quad M' \equiv N'}{N \xrightarrow{\alpha} N'}$	
MOBILITY(I)	$\dfrac{}{M \mid P{:}G \xrightarrow{\mu} M \mid P{:}G'}$	$G' \neq G$, $G' \subseteq G \cup fgn(M)$, $I(M \mid P{:}G) \implies I(M \mid P{:}G')$
PAR(I)	$\dfrac{M \xrightarrow{\alpha} M'}{M \mid N \xrightarrow{\alpha} M' \mid N}$	$bn(\alpha) \cap fn(N) = \emptyset$ $I(M \mid N) \implies I(M' \mid N)$
COM	$\dfrac{M \xrightarrow{\overline{G}x} M' \quad N \xrightarrow{G'(y)} N'}{M \mid N \xrightarrow{\overline{G}x} M' \mid N'\{x/y\}}$	$G \cap G' \neq \emptyset$
GNAME-RES1	$\dfrac{M \xrightarrow{\alpha} M'}{(\nu\, g)M \xrightarrow{\alpha \setminus \{g\}} (\nu\, g)M'}$	$\alpha \in \{\tau, \mu\}$, or $gn(\alpha) \setminus \{g\} \neq \emptyset$
GNAME-RES2	$\dfrac{M \xrightarrow{\overline{G}x} M'}{(\nu\, g)M \xrightarrow{\tau} (\nu\, g)M'}$	$G = \{g\}$

Table 3: Transition rules for ω_0-calculus structured node expressions.

and GNAME-RES2 together mean that a local-broadcast send is non-blocking; i.e., it can be performed on a set of restricted groups even when there are no corresponding receive actions. In contrast, other actions containing gnames, such as local-broadcast receive, are not covered by GNAME-RES2, and hence have blocking semantics: a system cannot perform actions involving restricted gnames unless there is a corresponding synchronizing action.

Comment (1e) Done. Explain why local-broadcast is non-blocking but receive is not.

In contrast to the broadcast calculi of [5, 13], a node that is capable of receiving a local broadcast is not forced to synchronize with the sender. The semantics of local broadcast in the ω-calculus allows a receiver to ignore a local-broadcast event even if this node is in the transmission range of the broadcasting node. A semantics of this nature captures the lossy transmission inherent in MANETs. The semantics of local broadcast can be modified to force all potential receivers to receive a local broadcast, as done in other broadcast calculi [5, 13]. This would require the addition of a side-condition to the PAR rule, allowing autonomous broadcast/receive actions only when the context (node expression N in the PAR rule) is incapable of synchronizing with that action.

Comment (1b) Done. Semantics of local broadcast to force non-lossy broadcast.

The notion of structural congruence (Table 1) considered in rule STRUCT is defined for processes (rules P1-P3) in the standard way—P and Q are structurally congruent if they are alpha-equivalent or congruent under the associa-

MOBILITY

$$(P_1:\{g_1,g_2\} \mid P_2:\{g_1\} \mid P_4:\{g_1,g_3\}) \mid P_3:\{g_2\} \xrightarrow{\mu} (P_1:\{g_1,g_2\} \mid P_2:\{g_1\} \mid P_4:\{g_1,g_3\}) \mid P_3:\{g_3\}$$

STRUCT

$$P_1:\{g_1,g_2\} \mid P_2:\{g_1\} \mid P_3:\{g_2\} \mid P_4:\{g_1,g_3\} \xrightarrow{\mu} P_1:\{g_1,g_2\} \mid P_2:\{g_1\} \mid P_3:\{g_3\} \mid P_4:\{g_1,g_3\}$$

GNAME-RES1 (thrice)

$$(\nu g_1)(\nu g_2)(\nu g_3)(P_1:\{g_1,g_2\} \mid P_2:\{g_1\} \mid P_3:\{g_2\} \mid P_4:\{g_1,g_3\}) \xrightarrow{\mu} (\nu g_1)(\nu g_2)(\nu g_3)(P_1:\{g_1,g_2\} \mid P_2:\{g_1\} \mid P_3:\{g_3\} \mid P_4:\{g_1,g_3\})$$

STRUCT

$$(\nu g_1)(\nu g_2)(P_1:\{g_1,g_2\} \mid P_2:\{g_1\} \mid P_3:\{g_2\} \mid (\nu g_3)(P_4:\{g_1,g_3\})) \xrightarrow{\mu} (\nu g_1)(\nu g_2)(P_1:\{g_1,g_2\} \mid P_2:\{g_1\} \mid (\nu g_3)(P_3:\{g_3\} \mid P_4:\{g_1,g_3\}))$$

STRUCT

$$(\nu g_1)(\nu g_2)(P_1:\{g_1,g_2\} \mid P_2:\{g_1\} \mid P_3:\{g_2\} \mid P_4:\{g_1\}) \xrightarrow{\mu} (\nu g_1)(\nu g_2)(P_1:\{g_1,g_2\} \mid P_2:\{g_1\} \mid (\nu g_3)(P_3:\{g_3\} \mid P_4:\{g_1,g_3\}))$$

Figure 3: Derivation for movement of N_3 from its position in Fig. 1 to that in Fig. 2.

tivity and commutativity of the choice ('+') operator—and then lifted to nodes (rules N1-N9). Two basic node expressions are structurally congruent if they have identical process interfaces and run structurally congruent processes (rule N8). Rules N4-N6 are for restriction on gnames. Rule N9 allows basic nodes to create and acquire a new group name or drop a local group name. Structural congruence of nodes includes alpha-equivalence (rule N7) and the associativity and commutativity of the parallel ('|') operator (rules N2 and N3).

Semantics of mobility. The semantics of node movement is defined by the MOBILITY rule, which states that the process interface of node $P:G$ can change from G to G' whenever the node is in parallel with another node M. In particular, the side condition $G' \subseteq G \cup fgn(M)$ stipulates that P may drop gnames from its interface or acquire free gnames from M.

The MOBILITY rule reflects the fact that P's interface may change when node $P:G$, or the nodes around it, are in motion. A change in P's interface may further result in a corresponding change in the overall network topology. Note that the rule does not specify which nodes moved, only that the topology has been updated as the result of movement of one or more nodes. The third side

Comment (2i) Done. "rule MOBILITY: checking the mobility invariant only when rule mobility is applied is not enough, since then rule PAR may allow to break the invariant (not all invariants are preserved by parallel composition)"

Comment (1f) related to comment (1n) about case of mobility in the congruence proof case 8 (Done in our REMARKS). Address the comment that "in MOBILITY(I) a node cannot connect to a node

condition to the MOBILITY rule, decrees that whenever $M \xrightarrow{\mu} M'$ is derived using the MOBILITY rule, the resulting transition must preserve a connectivity invariant.

We thus have that the MOBILITY and PAR rules in particular, and the calculus's semantics in general, are parameterized by the connectivity invariant, thus taking into account the constraints on node movement.

An example derivation of node movement is shown in Figure 3. This derivation was obtained using the structural congruence and transition rules defining the semantics of the ω-calculus, and "connectedness" as the connectivity invariant.

3.3. The ω_1-Calculus

The ω_1- and ω_2-calculi are defined in a modular fashion by adding new syntactic constructs, and associated inference rules for their semantics, to the ω_0-calculus. In this subsection, we consider the extension ω_1.

Extending ω_0 ***to*** ω_1**.** Syntactically, we obtain ω_1 from ω_0 as follows:

- We add restriction operators for *pnames* for both process-level and node-level expressions. We use the standard notation of $(\nu x)P$ for a pname x restricted to a process expression P, and $(\nu x)N$ for a pname x restricted to a node expression N. As usual, x is bound in $(\nu x)P$ and $(\nu x)N$.
- We introduce unicast communication as a prefix operator for process expressions. Although unicast in principle can be implemented on top of broadcast, we prefer to give it first-class status, as it is a frequent action in MANET protocols. Doing so also facilitates concise modeling and deterministic reasoning (only the intended recipient can receive a unicast message). We use the standard notation of $\overline{x}y$ to denote the sending of name y along x, and $x(y)$ to denote the reception of a name along x that will bind to y. As usual, x and y are free in the expression $\overline{x}y.P$, and x is free and y is bound in $x(y).P$.

Unlike in ω_0 where *pnames* are used strictly as data values, in ω_1, *pnames* (the set **Pn**) can be used as communicable data as well as communication (unicast) channels.

Semantically, the introduction of scoped pnames needs new inference rules to handle scope extrusion. We add OPEN and CLOSE rules (as in the π-calculus [12]) and, in addition to the broadcast communication rule (COM) of ω_0, a rule for communication of bound names. We also add RES rules at the process and node levels to disallow communication over a restricted name. These additional rules follow closely the standard rules for handling scopes and scope extrusion in the π-calculus; details are omitted. New structural congruence rules are added to take the restriction of pnames into account. For instance, restriction of pnames and gnames commute (i.e. $(\nu x)(\nu g)N \equiv (\nu g)(\nu x)N$), and the restriction operator can be pushed into or pulled out of node and process expressions as long as free names are not captured. At first glance, it may appear that the structural congruence rules for scope extension of pnames are redundant in the presence of the scope-extrusion rules (OPEN/CLOSE). However, the

Rule Name	Rule	Side Condition
UNI-SEND	$\dfrac{}{(\overline{z}x.P){:}G \xrightarrow{\overline{z{:}G}x} P{:}G}$	$G \neq \emptyset$
UNI-RECV	$\dfrac{}{(z(x).P){:}G \xrightarrow{z{:}G(x)} P{:}G}$	$G \neq \emptyset$
UNI-COM	$\dfrac{M \xrightarrow{\overline{z{:}G}x} M' \quad N \xrightarrow{z{:}G'(y)} N'}{M \mid N \xrightarrow{\tau} M' \mid N'\{x/y\}}$	$G \cap G' \neq \emptyset$

Table 4: Transition rules for unicast communication in ω_1-calculus.

OPEN/CLOSE rules are essential for reasoning about open systems, and the scope extension rules are essential for defining normal forms (see Definition 3).

The addition of unicast communication raises certain interesting issues with respect to mobility. Recall that *groups* encapsulate the locality of a process. When two processes share a private name, they can use that name as a channel of communication. However, after establishing that link, if the processes move away from each other, they may no longer be able to use that name as a channel. In summary, unicast channels should also respect the locality of communication. We enforce this in the ω_1-calculus by annotating unicast action labels with the interfaces of the participating processes, and allowing synchronization between actions only when their interfaces overlap (meaning that the processes are in each other's transmission range). Hence, the execution of a unicast send action of value x on channel z by a basic node with process interface G is represented by action label $\overline{z{:}G}x$; the corresponding receive action is labeled $z{:}G(x)$.

The semantic rules for unicast send (UNI-SEND), receive (UNI-RECV), and synchronization (UNI-COM) are given in Table 4. Scope extrusion via unicast communication is accomplished by naturally extending their π-calculus counterparts (OPEN/CLOSE) rules as follows. Bound-output actions (due to OPEN) are annotated with the interface of the participating process, and the CLOSE rule applies only when the interfaces overlap. These extensions are straightforward, and the details are omitted.

Comment (2j) Done. "I will present P rules before N rules"

The set of bound names and gnames for the transition labels introduced by the ω_1-calculus are given below:
$bn(\overline{z:G}x) = \emptyset$, $bn(z:G(x)) = \{x\}$, $bn((\nu x)\overline{z:G}x) = \{x\}$, $bn((\nu x)\overline{G}x) = \{x\}$.
$gn(\overline{z:G}x) = G$, $gn(z:G(x)) = G$, $gn((\nu x)\overline{z:G}x) = G$, $gn((\nu x)\overline{G}x) = G$.

Done. Define *bn* for transition labels due to unicast

Note that the scope of a name may encompass different processes regardless of their interfaces, and hence two processes may share a secret even when they are outside each others transmission ranges. The restriction we impose is that shared names can be used as unicast channels only when the processes are within each others transmission ranges.

P4.	$(\nu x)P \equiv P$, if $x \notin fn(P)$
P5.	$(\nu x)(\nu y)P \equiv (\nu y)(\nu x)P$
P6.	$P \mid Q \equiv Q \mid P$
P7.	$(P \mid Q) \mid R \equiv P \mid (Q \mid R)$
P8.	$(\nu x)P_1 \mid P_2 \equiv (\nu x)(P_1 \mid P_2)$ if $x \notin fn(P_2)$
N10.	$(\nu x)M \equiv M$, if $x \notin fpn(M)$
N11.	$(\nu x)M_1 \mid M_2 \equiv (\nu x)(M_1 \mid M_2)$, if $x \notin fpn(M_2)$
N12.	$(\nu x)(\nu y)M \equiv (\nu y)(\nu x)M$
N13.	$(\nu g)(\nu x)M \equiv (\nu x)(\nu g)M$
N14.	$((\nu x)P):G \equiv (\nu x)(P:G)$

Table 5: Additional structural congruence rules for the ω-node expressions.

Rule Name	Rule	Side Condition
PROC-PAR	$\dfrac{P:G \xrightarrow{\alpha} P':G}{(P \mid Q):G \xrightarrow{\alpha} (P' \mid Q):G}$	$bn(\alpha) \cap fn(Q) = \emptyset$
PROC-COM	$\dfrac{P:G \xrightarrow{\overline{z:G}x} P':G \quad Q:G \xrightarrow{z:G(y)} Q':G}{(P \mid Q):G \xrightarrow{\tau} (P' \mid Q'\{x/y\}):G}$	
PROC-CLOSE	$\dfrac{P:G \xrightarrow{(\nu x)\overline{z:G}x} P':G \quad Q:G \xrightarrow{z:G(x)} Q':G}{(P \mid Q):G \xrightarrow{\tau} ((\nu x)(P' \mid Q')):G}$	

Table 6: Additional transitional semantics rules for basic ω-node expressions.

3.4. The full ω-calculus: ω_2-calculus.

We obtain the ω_2-calculus by adding the parallel composition ('|') operator at the process level, thereby allowing concurrent processes within a node. This addition facilitates e.g. the modeling of communication between layers of a protocol stack running at a single node; it also renders the π-calculus a subcalculus of the ω_2-calculus. In ω_2, the actions of two processes within a node may be interleaved. Moreover, two processes within a node can synchronize using unicast (binary) communication. We add PAR, COM and CLOSE rules corresponding to intra-node interleaving, synchronization and scope extrusion, respectively; these rules are straightforward extensions of the corresponding rules in the π-calculus.

The syntax of processes in the ω-calculus is defined by the following gram-

Comment (2k) Done. "I don't like the use of the same primitive for intra-node and inter-node communication: these correspond to very different physical operations (a wireless communication and an internal synchronization); I would like at least a more detailed discussion about this; also, I'm not convinced that becoming a conservative extension of the pi-calculus is an interesting property, since the kind of systems to be modeled is very

Rule Name	**Rule**	**Side Condition**
UNI-OPEN	$\dfrac{M \xrightarrow{\overline{z{:}G}x} M'}{(\nu x)M \xrightarrow{(\nu x)\overline{z{:}G}x} M'}$	$x \neq z$
UNI-CLOSE	$\dfrac{M \xrightarrow{(\nu x)\overline{z{:}G}x} M' \quad N \xrightarrow{z{:}G'(x)} N'}{M \mid N \xrightarrow{\tau} (\nu x)(M' \mid N')}$	$G \cap G' \neq \emptyset$
OPEN	$\dfrac{M \xrightarrow{\overline{G}x} M'}{(\nu x)M \xrightarrow{(\nu x)\overline{G}x} M'}$	
COM-RES	$\dfrac{M \xrightarrow{(\nu x)\overline{G}x} M' \quad N \xrightarrow{G'(x)} N'}{M \mid N \xrightarrow{(\nu x)\overline{G}x} M' \mid N'}$	$G \cap G' \neq \emptyset$
CLOSE	$\dfrac{M \xrightarrow{(\nu x)\overline{G}x} M'}{(\nu\, g)M \xrightarrow{\tau} (\nu\, g)(\nu x)M'}$	$G = \{g\}$
PNAME-RES	$\dfrac{M \xrightarrow{\alpha} M'}{(\nu x)M \xrightarrow{\alpha} (\nu x)M'}$	$x \notin n(\alpha)$

Table 7: Additional transition semantics rules for structured ω-node expressions.

mar:

$$\begin{array}{rcl} P & ::= & nil \mid Act.P \mid P+P \mid (\nu x)P \mid [x=y]P \mid P|P \mid A(\vec{x}) \\ Act & ::= & \overline{x}y \mid x(y) \mid \overline{\mathbf{b}}x \mid \mathbf{r}(x) \mid \tau \end{array}$$

The following grammar defines the syntax of node expressions in the ω-calculus:

$$M \quad ::= \quad \mathbf{0} \mid P{:}G \mid (\nu g)M \mid (\nu x)M \mid M|M$$

The structural congruence rules for the ω-calculus are given in Tables 1 and 5, and the transitional semantics rules are given in Tables 2, 3, 4, 6, and 7.

4. Bisimulation, Congruence Results and Other Properties of the ω-Calculus

In this section, we prove some fundamental properties of the ω-calculus, including congruence results for strong bisimulation equivalence and a weak version of bisimulation equivalence that treats τ- and μ-actions as unobservable.

Embedding of the π-Calculus. The ω-calculus is a conservative extension of the π-calculus [12]. That is, every process expression P in the π-calculus can be translated to an ω-node expression $P : G$, for $G \subseteq \mathbf{Gn}$ and $G \neq \emptyset$, such that the transition system generated by $P : G$ is isomorphic to the one generated by P. We impose the condition $G \neq \emptyset$ since a basic node with an empty interface ($P : \{\}$) cannot perform any action. This property is formally stated by the following theorem, which is readily proved by induction on the length of derivations.

Comment (1h). "If interfaces may be empty, why not let P map to $P{:}\{\}$ in Thm 1".

Theorem 1. *For any process expression P in the π-calculus, $P : G$ is a node expression in the ω-calculus, where $G \subseteq \mathbf{Gn}$ and $G \neq \emptyset$. Moreover, $P \xrightarrow{\alpha} P'$ is a transition derivable from the operational semantics of the π-calculus if and only if $P : G \xrightarrow{\alpha'} P' : G$ is derivable from the operational semantics of the ω-calculus, and one of the following conditions hold: (i) $\alpha = \alpha' = \tau$; (ii) $\alpha = x(y)$ and $\alpha' = x : G(y)$; (iii) $\alpha = \overline{x}y$ and $\alpha' = \overline{x{:}G}y$; or (iv) $\alpha = (\nu y)\overline{x}y$ and $\alpha' = (\nu y)\overline{x{:}G}y$, for some names x, y.*

Done: **Pn** and **Gn** are disjoint, so the requirement that 'g is a fresh group name not in P' can be removed.

Decidability of the Finite-Control Fragment. The *finite-control* fragment of the ω-calculus, as in the case of the π-calculus, is the subcalculus where recursive definitions do not contain the parallel operator ('|') and every occurrence of process identifiers is guarded. Reachability properties are decidable for closed process expressions (i.e. those without free names) specified in the finite-control fragment [4]. We extend the notion of finite control to the ω-calculus, and show that reachability remains decidable for closed node expressions. This result, formally stated in Theorem 2, is of practical importance in verifying MANET system specifications. Formally, we say that an ω-calculus expression N is *reachable* from M (denoted by $M \longrightarrow^* N$) if there is a finite sequence of transitions $M \xrightarrow{\alpha_1} M_1 \xrightarrow{\alpha_2} M_2 \cdots \xrightarrow{\alpha_n} N$.

Theorem 2. *Let M be a closed finite-control ω-calculus expression and let $R_M = \{N \mid M \longrightarrow^* N\}_{\equiv}$ be the set of node expressions reachable from M modulo the structural congruence relation $\equiv$. Then, R_M is finite.*

The following proof uses the finite reachability result for the finite-control π-calculus given in [4].

Proof Sketch: Consider the fragment ω_π of the ω-calculus without broadcast actions and the MOBILITY rule. For a node expression M in ω_π, the corresponding π-calculus process expression, denoted by M_π, is obtained from M by deleting process interfaces and gname restrictions. Let M be a ω_π-expression such that all process expressions have the same process interface. Then M's transition system is isomorphic to that of M_π.

Now further assume that M is closed and finite-control. Then the set of expressions reachable from M, R_M, is similar (except for occurrences of process interfaces and gnames) to that for M_π. Since only finitely many expressions are reachable from M_π, R_M is also finite.

Next, extend ω_π to $\omega_{b\pi}$ by including broadcast actions. Let M_1 be such a $\omega_{b\pi}$-expression that is both closed and finite-control. The process contexts due to broadcast action prefixes are analyzed in a similar manner as the binary-synchronization action prefixes. Using an argument similar to the one used above for ω_π, it can be concluded that R_{M_1} is finite.

Finally, we include the MOBILITY rule in $\omega_{b\pi}$, extending $\omega_{b\pi}$ to the ω-calculus. Let M_2 be a closed finite-control ω-expression. The MOBILITY rule affects only the gnames (including process-interfaces) appearing in expressions reachable from M_2. It can be observed that the set R_{M_2} is a variant of the set R_{M_1}, with different combinations of process-interfaces (permitted by the MOBILITY rule) attached to the process expressions appearing in the elements of R_{M_1}. The different combinations of process interfaces possible for n basic nodes in an ω-expression (modulo $\equiv$) is finite and bounded by the number of topologies that a network of n nodes can assume. This implies that R_{M_2} is finite.

Hence, reachability for the finite-control fragment of the ω-calculus is decidable. □

Comment (2l) Done. "you should state this assumption before or in the theorem statement, not in the proof". Removed the assumption, see the response in file review2.txt.

Bisimulation for the ω-Calculus. The definition of strong (late) bisimulation for the π-calculus [12] can be extended to the ω-calculus.

Comment (2m) Done. "why do you choose late bisimulation?"

Definition 1. *A relation $S \subseteq \mathbf{N} \times \mathbf{N}$ is a* strong simulation *if $M\,S\,N$ implies:*

- *fgn(M) = fgn(N), and*
- *whenever $M \xrightarrow{\alpha} M'$ where $bn(\alpha)$ is fresh then:*
  - *if $\alpha \in \{G(x),\, z{:}G(x)\}$, there exists an N' s.t. $N \xrightarrow{\alpha} N'$ and for all $y \in$ **Pn**, $M'\{y/x\}\,S\,N'\{y/x\}$,*
  - *if $\alpha \notin \{G(x),\, z{:}G(x)\}$, there exists an N' s.t. $N \xrightarrow{\alpha} N'$ and $M'\,S\,N'$.*

S is a strong bisimulation *if both S and S^{-1} are strong simulations. Nodes M and N are* strong bisimilar*, written $M \sim N$, if $M\,S\,N$ for some* strong bisimulation *S.*

Proposition 3. *(i) $\sim$ is an equivalence; and (ii) $\sim$ is the largest strong bisimulation.*

Strong bisimulation equivalence is a congruence for the ω-calculus, as formally stated in Theorem 9. We use the *bisimulation up to* $\equiv$ technique [19] to establish this result. The following definitions and lemmas are also needed.

Notation: For a given relation R, the relation $\equiv R \equiv$ is given by: $\{(x, y) \mid (x', y') \in R, x \equiv x', y \equiv y'\}$.

Definition 2. *A symmetric relation $S \subseteq \mathbf{N} \times \mathbf{N}$ is a* strong bisimulation up to $\equiv$ *if $M\, S\, N$ implies*

- *$fgn(M) = fgn(N)$, and*
- *whenever $M \xrightarrow{\alpha} M'$ where $bn(\alpha)$ is fresh then:*
  - *if $\alpha \in \{G(x),\, z{:}G(x)\}$, there exists an N' s.t. $N \xrightarrow{\alpha} N'$ and for all $y \in \boldsymbol{Pn}$, $M'\{y/x\} \equiv S \equiv N'\{y/x\}$,*
  - *if $\alpha \notin \{G(x),\, z{:}G(x)\}$, there exists an N' s.t. $N \xrightarrow{\alpha} N'$ and $M' \equiv S \equiv N'$.*

Lemma 4. *If S is a strong bisimulation up to $\equiv$, then for any $M, N \in \mathbf{N}$, $M\, S\, N$ implies $M \sim N$.*

Proof: For any $M, N \in \mathbf{N}$, $M\, S\, N$ implies $M \equiv S \equiv N$. It is sufficient to show that $\equiv S \equiv$ is a strong bisimulation because then $M \equiv S \equiv N$ would imply that M and N are strong bisimilar. $M \equiv S \equiv N$ implies there exist some M_1 and N_1 s.t. $M \equiv M_1$, $N \equiv N_1$, and $M_1\, S\, N_1$. From STRUCT rule, $M \xrightarrow{\alpha} M'$ implies that there exists M_1' s.t. $M_1 \xrightarrow{\alpha} M_1'$ and $M' \equiv M_1'$.

For the case $\alpha \notin \{G(x), z{:}G(x)\}$, using Def. 2 it can be inferred that $M_1\, S\, N_1$ and $M_1 \xrightarrow{\alpha} M_1'$ imply that there exists N_1' s.t. $N_1 \xrightarrow{\alpha} N_1'$ and $M_1' \equiv S \equiv N_1'$. $M\, S\, N$ and $M \xrightarrow{\alpha} M'$ imply that there exists N' s.t. $N \xrightarrow{\alpha} N'$, and $N_1' \equiv N'$ because $N \equiv N_1$. $M_1' \equiv S \equiv N_1'$ holds since $M_1\, S\, N_1$ and S is a strong bisimulation up to $\equiv$. By transitivity of $\equiv$, $M' \equiv S \equiv N'$.

Similarly, it can be shown that for $\alpha \in \{G(x), z{:}G(x)\}$, and for each $y \in \mathbf{Pn}$, $M'\{y/x\} \equiv S \equiv N'\{y/x\}$.

From Def. 2 and $M\, S\, N$, it holds that $fgn(M) = fgn(N)$.

Hence, $\equiv S \equiv$ is a strong bisimulation. Therefore, $M \equiv S \equiv N$ implies $M \sim N$. □

An intermediate step in establishing that strong bisimulation equivalence is a congruence for the ω-calculus is to prove it for ω-expressions in a normal form, defined below. We use the term "guarded restrictions" in the context of ω-expressions to mean restrictions that are preceded by an action prefix.

Definition 3 (Normal Form). *An ω-expression is in* normal form *if all bound names are distinct and all unguarded restrictions are at the top level with all gnames preceding pnames.*

We use $\mathbf{N_{nf}}$ to denote the set of ω-node expressions in normal form. The structural congruence rules are extended by the following rules (as in [15]).

$(\nu x)\mathbf{0} \equiv \mathbf{0}$

$(\nu x)P + Q \equiv (\nu x)(P + Q) \qquad if\, x \notin fn(Q)$

$[y = z](\nu x)P \equiv (\nu x)[y = z]P \qquad if\, x \neq y \text{ and } x \neq z$

$A(\vec{y}) \equiv P\{\vec{y}/\vec{x}\} \qquad A(\vec{x}) \stackrel{def}{=} P$

Comment (2n) Done. "can the restrictions at top level be referred to names that do not occur in the term? Otherwise Lemma 7 becomes false"

Comment (2o) Done. "I will put the extended structural congruence before the proposition, otherwise the proposition is false"

Proposition 5. *Every ω-expression is structurally congruent to an ω-expression in normal form.*

Every ω-expression can be converted into a structurally congruent ω-expression in normal form by renaming all bound names so that they are distinct and using structural congruence rules to pull out all unguarded restrictions to the outermost level.

The following lemma originally appeared in [15] and is lifted here to the ω-calculus.

Done: Lemma 6 and lemma 7 should have N' and M'' respectively existentially quantified.

Lemma 6. *If $M \xrightarrow{\alpha} M'$ and $M \equiv N$ where $N \in \mathbf{N_{nf}}$, then there exists $N' \in \mathbf{N}$ such that by inference of no greater depth, $N \xrightarrow{\alpha} N'$ and $M' \equiv N'$.*

Proof Idea: The proof is by induction on the inference of $M \equiv N$ and involves examination of all the structural congruence rules. □

Lemma 7. *For every $M, M' \in \mathbf{N}$, if $M \xrightarrow{\alpha} M'$, then there exists an $N \in \mathbf{N_{nf}}$ such that $N \equiv M$ and*

(i) N is of the form $(\nu\tilde{g})(\nu\tilde{x})N'$ where $\tilde{g}$ and $\tilde{x}$ are nonempty sets, and

(ii) there exists $M'' \in \mathbf{N}$ such that $N \xrightarrow{\alpha} M''$, $M'' \equiv M'$, and $N \xrightarrow{\alpha} M''$ can be derived without using STRUCT rules in the last two steps of the derivation.

Proof Sketch: Clearly, we can always find an N in normal form obeying condition (i) that is equivalent to any given $M \in \mathbf{N}$. Since N has an outermost (non-empty) gname restriction and a non-empty pname restriction at the next level, any derivation for a transition from N will involve at least two steps.

Consider the shortest derivation for $N \xrightarrow{\alpha} M''$ (shortest among all N equivalent to M). For such a derivation, the last step cannot be an application of STRUCT rule. To the contrary, assume that the last step in the derivation is an application of the STRUCT rule. Then the last step is of the form:

$$\frac{N \equiv M_1 \quad M_1 \xrightarrow{\alpha} M_2 \quad M_2 \equiv M}{N \xrightarrow{\alpha} M}$$

M_1 cannot be in normal form; otherwise there is a shorter derivation. However, by Lemma 6, there is a normal form equivalent to M_1 that has at least as short a derivation. Thus, there is a shorter derivation for $N'' \xrightarrow{\alpha} M''$ for some normal form $N'' \equiv M$, which is a contradiction. Hence the last step in the shortest derivation cannot be an application of the STRUCT rule.

This means that the last step in the shortest derivation must be due to the outermost gname restriction. We can similarly argue that in the shortest derivation, the next-to-last step is not an application of STRUCT rule. □

Lemma 8. *For all $M_1, M_2 \in \mathbf{N_{nf}}$, i.e., M_1, M_2 are in normal form, the following hold:*

(i) $M_1 \sim M_2$ implies $\forall x \in \mathbf{Pn} : (\nu x)M_1 \sim (\nu x)M_2$;

(ii) $M_1 \sim M_2$ implies $\forall g \in \mathbf{Gn} : (\nu g)M_1 \sim (\nu g)M_2$; and

(iii) $M_1 \sim M_2$ implies $\forall N \in \mathbf{N_{nf}}$: $M_1|N \sim M_2|N$.

Proof Sketch. We give a sketch of the proof in what follows. The complete proof is given in Appendix A.

We show parts *(i–iii)* of the lemma simultaneously by considering the set $S = \{((\nu\tilde{g})(\nu\tilde{x})(M_1|N), (\nu\tilde{g})(\nu\tilde{x})(M_2|N)) \mid M_1 \sim M_2, \tilde{g} \subseteq \mathbf{Gn}, \tilde{x} \subseteq \mathbf{Pn}, M_1, M_2, N \in \mathbf{N_{nf}}\}$. Following Lemma 4, it is sufficient to show that S is a strong bisimulation upto $\equiv$.

Note that if $M_1 \sim M_2$ then $fgn(M_1) = fgn(M_2)$, and hence $fgn((\nu\tilde{g})(\nu\tilde{x})(M_1|N)) = fgn((\nu\tilde{g})(\nu\tilde{x})(M_2|N))$ for all $\tilde{g}, \tilde{x}$ and N. We then show that every transition from $(\nu\tilde{g})(\nu\tilde{x})(M_1|N)$ can be matched by $(\nu\tilde{g})(\nu\tilde{x})(M_2|N)$ by considering the derivations of transitions. Transitions for $(\nu\tilde{g})(\nu\tilde{x})(M_1|N)$ can be derived by use of rules CLOSE, GNAME-RES1, GNAME-RES2, MOBILITY, PAR, UNI-COM, UNI-CLOSE, COM, COM-RES, UNI-OPEN, OPEN and PNAME-RES. Only the last three steps of each transition derivation are considered in the proof. Most importantly, following Lemma 7, we do not need to consider derivations that use STRUCT rules in the last two steps. From the structural operational semantics, the last step of a derivation will be due to the outermost $(\nu\tilde{g})$ in the expression, the next-to-last step will be due to the $(\nu\tilde{x})$ following the outermost $(\nu\tilde{g})$, and the step before that will be due to the parallel composition $(M_1|N)$. We omit in the proof the symmetric cases arising due to the commutativity of the parallel operator '|'. This gives rise to 15 cases (owing to the combinations of rules applied during the last three steps in a derivation). For illustration, we show here one such case:

Comment (1i) Done. Is reference to Lemma 7, possible since according to the definition of set S it is not known that $\tilde{x}$ and $\tilde{g}$ are non-empty?

Case CLOSE, OPEN, COM:

$(\nu\tilde{g})(\nu\tilde{x})(M_1|N) \xrightarrow{\tau} (\nu\tilde{g})(\nu\tilde{x})(M_1'|N'\{x'/y\})$ given $M_1 \xrightarrow{\overline{G}x'} M_1'$ and $N \xrightarrow{G'(y)} N'$. The derivation is as follows, where $\tilde{x}_1 = \tilde{x} \setminus \{x'\}$.

$$\text{COM:}\quad \frac{M_1 \xrightarrow{\overline{G}x'} M_1' \quad N \xrightarrow{G'(y)} N'}{M_1|N \xrightarrow{\overline{G}x'} M_1'|N'\{x'/y\}} \qquad G \cap G' \neq \emptyset$$

$$\text{OPEN:}\quad \frac{(\nu\tilde{x})(M_1|N) \xrightarrow{(\nu x')\overline{G}x'} (\nu\tilde{x_1})(M_1'|N'\{x'/y\})}{(\nu\tilde{g})(\nu\tilde{x})(M_1|N) \xrightarrow{\tau} (\nu\tilde{g})(\nu x')(\nu\tilde{x}_1)(M_1'|N'\{x'/y\})} \qquad G \setminus \tilde{g} = \emptyset$$

Since $M_1 \sim M_2$, $M_1 \xrightarrow{\overline{G}x'} M_1'$ means that there is an M_2' such that $M_2 \xrightarrow{\overline{G}x'} M_2'$ and $M_1' \sim M_2'$. Moreover, there exist expressions M_{N1}', M_{N2}' and N_N' in normal form such that $M_1' \equiv M_{N1}'$, $M_2' \equiv M_{N2}'$ and $N'\{x'/y\} \equiv N_N'$. Now, since $M_1' \sim M_2'$, we know $M_{N1}' \sim M_{N2}'$. Hence by construction of S, we can conclude that the pair $((\nu\tilde{g})(\nu x')(\nu\tilde{x}_1)(M_{N1}'|N_N'), (\nu\tilde{g})(\nu x')(\nu\tilde{x}_1)(M_{N2}'|N_N')) \in S$, and hence $((\nu\tilde{g})(\nu\tilde{x})(M_1'|N'\{x'/y\}), (\nu\tilde{g})(\nu\tilde{x})(M_2'|N'\{x'/y\})) \in \equiv S \equiv$.

By considering the 15 cases that cover all possible derivations, we conclude that for every transition from $(\nu\tilde{g})(\nu\tilde{x})(M_1|N)$, there is a transition from $(\nu\tilde{g})(\nu\tilde{x})(M_2|N)$ such that the destinations of the two transitions are related by $\equiv S \equiv$. Thus we establish that S is a strong bisimulation upto $\equiv$. □

Comment (2p) Done. "it would be interesting to know also what happens by adding behavior to nodes"

Theorem 9 (Congruence). *$\sim$ is a congruence for the ω-calculus; i.e., for all $M_1, M_2 \in \mathbf{N}$, the following hold:*

(i) $M_1 \sim M_2$ implies $\forall x \in \mathbf{Pn} : (\nu x)M_1 \sim (\nu x)M_2$;

(ii) $M_1 \sim M_2$ implies $\forall g \in \mathbf{Gn} : (\nu g)M_1 \sim (\nu g)M_2$; and

(iii) $M_1 \sim M_2$ *implies* $\forall N \in \mathbf{N} : M_1|N \sim M_2|N$.

Proof: Let $M_1 \equiv M_{N_1}$ and $M_2 \equiv M_{N_2}$, where $M_1, M_2 \in \mathbf{N}$ and $M_{N_1}, M_{N_2} \in \mathbf{N_{nf}}$. Then the following hold:

- $M_1 \sim M_2$ implies $M_{N_1} \sim M_{N_2}$ (from Definition 2 and Lemma 4). $M_{N_1} \sim M_{N_2}$ implies $\forall x \in \mathbf{P_n}$: $(\nu x)M_{N_1} \sim (\nu x)M_{N_2}$ (by Lemma 8), which in turn implies $(\nu x)M_1 \sim (\nu x)M_2$ (from Definition 2 and Lemma 4). Therefore, whenever $M_1 \sim M_2$ then $(\nu x)M_1 \sim (\nu x)M_2$.
- $M_1 \sim M_2$ implies $M_{N_1} \sim M_{N_2}$ (from Definition 2 and Lemma 4). $M_{N_1} \sim M_{N_2}$ implies $\forall g \in \mathbf{G_n}$: $(\nu g)M_{N_1} \sim (\nu g)M_{N_2}$ (by Lemma 8), which in turn implies $(\nu g)M_1 \sim (\nu g)M_2$ (from Definition 2 and Lemma 4). Therefore, whenever $M_1 \sim M_2$ then $(\nu g)M_1 \sim (\nu g)M_2$.
- $M_1 \sim M_2$ implies $M_{N_1} \sim M_{N_2}$ (from Definition 2 and Lemma 4). $M_{N_1} \sim M_{N_2}$ implies for any $N \in \mathbf{N}$, and $N \equiv N_N$ where, $N_N \in \mathbf{N_{nf}}$: $(M_{N_1}|N_N) \sim (M_{N_2}|N_N)$ (by Lemma 8), which in turn implies $(M_1|N) \sim (M_2|N)$ (from Definition 2 and Lemma 4). Therefore, whenever $M_1 \sim M_2$ then $(M_1|N) \sim (M_2|N)$.

$\sim$ is preserved by all node contexts. Hence, $\sim$ is a congruence . □

Added about our choice for late semantics and late bisimulation.

Recall that we defined a late semantics for transition systems in the ω-calculus. Late semantics yields a more deterministic proof system for deriving transitions (as compared to that of early semantics) because of the late instantiation of an input name. Using the late semantics we defined a strong late bisimulation. Late bisimulation is more natural for auotmated verification tools because the late instantiation of names leads to more efficient checking of bisimulation.

Early bisimulation can also be defined for the ω-calculus using the late semantics. It turns out that early bisimulation equivalence is also a congruence for the ω-calculus. The fact that nodes in the ω-calculus cannot be placed in the context of an input or output prefix (only processes can be) makes the congruence result hold for both late and early bisimulation equivalences. In contrast for the π-calculus neither late nor early bisimulation equivalence is a congruence due to input prefix. The congruence results for early bisimulation equivalence in the ω-calculus can be established similar to that for the late bisimulation. A lemma similar to Lemma 8 can be used to prove congrunce for early bisimulation equivalence. In the proof of Lemma 8 given in Appendix A, all the cases except case 9 will be identical to those for late bisimulation equivalence. Case 9 which includes derivations for input transition labels, will have a more elaborate proof for early bisimulation equivalence to consider early instantiation of input names.

Weak Bisimulation for the ω-Calculus. We can also define a notion of *weak bisimulation* for the ω-calculus, in which τ- and μ-actions are treated as unobservable. Its definition is similar to that for strong bisimulation (Definition 1) and is given in Definition 4. We also establish that weak bisimulation equivalence, like its strong counterpart, is a congruence for the ω-calculus.

We use $\Longrightarrow$ to denote $\stackrel{(\tau|\mu)}{\longrightarrow}^*$, i.e., zero or more τ- or μ-transitions, and $\stackrel{\widehat{\alpha}}{\Longrightarrow}$

to denote $\xrightarrow{(\tau|\mu)^*} \xrightarrow{\alpha} \xrightarrow{(\tau|\mu)^*}$ if $\alpha \notin \{\tau, \mu\}$ and $\Longrightarrow$ otherwise.

Definition 4. *A relation $S \subseteq \mathbf{N} \times \mathbf{N}$ is a* weak simulation *if $M\,S\,N$ implies:*

- *fgn(M) = fgn(N), and*
- *whenever $M \xrightarrow{\alpha} M'$ where $bn(\alpha)$ is fresh then:*
  - *if $\alpha \in \{G(x),\, z{:}G(x)\}$, there exists an N'' s.t. $N \Longrightarrow \xrightarrow{\alpha} N''$ and for all $y \in \boldsymbol{Pn}$, there exists an N' s.t. $N''\{y/x\} \Longrightarrow N'$ and $M'\{y/x\}\,S\,N'$,*
  - *if $\alpha \notin \{G(x),\, z{:}G(x)\}$, there exists an N' s.t. $N \xRightarrow{\widehat{\alpha}} N'$ and $M'\,S\,N'$.*

S is a weak bisimulation *if both S and S^{-1} are weak simulations. Nodes M and N are* weak bisimilar*, written $M \approx N$, if $M\,S\,N$, for some* weak bisimulation *S.*

Proposition 10. *(i) $\approx$ is an equivalence relation; and (ii) $\approx$ is the largest weak bisimulation.*

Weak bisimulation equivalence is a congruence for the ω-calculus as formally stated below.

Comment (2q) Done. "I think you need closure under substitutions for the case of labels G(x) and z:G(x), which does not hold neither for pi-calculus neither here."

Theorem 11 (Congruence). *$\approx$ is a congruence for the ω-calculus; i.e., for all $M_1, M_2 \in \mathbf{N}$, the following hold:*

(i) $M_1 \approx M_2$ implies $\forall x \in \mathbf{Pn} : (\nu x)M_1 \approx (\nu x)M_2$;
(ii) $M_1 \approx M_2$ implies $\forall g \in \mathbf{Gn} : (\nu g)M_1 \approx (\nu g)M_2$; and
(iii) $M_1 \approx M_2$ implies $\forall N \in \mathbf{N} : M_1|N \approx M_2|N$.

Proof Sketch. It suffices to show that $S = \{((\nu\tilde{g})(\nu\tilde{x})(M_1|N),\, (\nu\tilde{g})(\nu\tilde{x})(M_2|N)) \,|\, M_1 \approx M_2, \tilde{g} \subseteq \mathbf{Gn}, \tilde{x} \subseteq \mathbf{Pn}, M_1, M_2, N \in \mathbf{N}\}$ is a weak bisimulation.

$M_1 \approx M_2$ implies that if $M_1 \xrightarrow{\alpha} M_1'$, where $\alpha \notin \{G(x),\, z : G(x)\}$, then there exists an M_2' such that $M_2 \xRightarrow{\widehat{\alpha}} M_2'$ and $M_1' \approx M_2'$. $M_2 \xRightarrow{\widehat{\alpha}} M_2'$ implies that there exist M_{2a} and M_{2b} such that $M_2 \Longrightarrow M_{2a}$, $M_{2a} \xrightarrow{\alpha} M_{2b}$, and $M_{2b} \Longrightarrow M_2'$.

It can be shown that if $(\nu\tilde{g})(\nu\tilde{x})(M_1|N) \xrightarrow{\alpha} (\nu\tilde{g})(\nu\tilde{x})(M_1'|N')$, then also $(\nu\tilde{g})(\nu\tilde{x})(M_{2a}|N) \xrightarrow{\alpha} (\nu\tilde{g})(\nu\tilde{x})(M_{2b}|N')$ which implies $(\nu\tilde{g})(M_2|N) \xRightarrow{\widehat{\alpha}} (\nu\tilde{g})(M_2'|N')$. We can similarly reason about the case when $\alpha \in \{G(x),\, z{:}G(x)\}$. Using these arguments along with the ideas used in the proof of congruence for strong bisimulation equivalence, it can be shown that weak bisimulation equivalence for ω-calculus is a congruence. □

5. Symbolic Semantics for the ω-Calculus

Comment (1j) Done. Better motivate the need for symbolic semantics

We now define a symbolic transitional semantics for the ω-calculus. The symbolic semantics binds names lazily and enables more efficient construction of transition systems for verification. In particular, transition system for a node expression is given using the traditional semantics (Section 3) assuming that the free names in the expression are all distinct. Consequently, transition system for an expression needs to be generated for each context in which the expression is used. In contrast, the symbolic semantics can be used to generate a symbolic transition system for each expression, and the transition system can then be

Comment (2r) Done. "It is not clear to me why do you need symbolic semantics for mismatch. Is it a standard requirement? Can you comment on this?"

Rule Name	**Rule**	**Side Condition**
MCAST	$\dfrac{}{(\overline{\mathbf{b}}x.P){:}G \xrightarrow{\mathbf{true},\overline{G}x} P{:}G}$	$G \neq \emptyset$
RECV	$\dfrac{}{(\mathbf{r}(x).P){:}G \xrightarrow{\mathbf{true},G(x)} P{:}G}$	$G \neq \emptyset$
CHOICE	$\dfrac{P{:}G \xrightarrow{\lambda} P'{:}G}{(P+Q){:}G \xrightarrow{\lambda} P'{:}G}$	
MATCH	$\dfrac{P{:}G \xrightarrow{C_1,\alpha} P'{:}G}{([x{=}y]P){:}G \xrightarrow{C_1 \wedge [x=y],\alpha} P'{:}G}$	$x, y \notin bn(\alpha)$
MISMATCH	$\dfrac{P{:}G \xrightarrow{C_1,\alpha} P'{:}G}{([x{\neq}y]P){:}G \xrightarrow{C_1 \wedge [x\neq y],\alpha} P'{:}G}$	$x, y \notin bn(\alpha)$
DEF	$\dfrac{P\{\vec{y}/\vec{x}\}{:}G \xrightarrow{\lambda} P'{:}G}{A(\vec{y}){:}G \xrightarrow{\lambda} P'{:}G}$	$A(\vec{x}) \stackrel{def}{=} P$

Table 8: Symbolic semantics for basic ω-node expressions.

Rule Name	Rule	Side Condition
UNI-SEND	$\dfrac{}{(\overline{z}x.P){:}G \xrightarrow{\mathbf{true},\overline{z{:}G}x} P{:}G}$	$G \neq \emptyset$
UNI-RECV	$\dfrac{}{(z(x).P){:}G \xrightarrow{\mathbf{true},z:G(x)} P{:}G}$	$G \neq \emptyset$
PROC-PAR	$\dfrac{P{:}G \xrightarrow{C,\alpha} P'{:}G}{(P \mid Q){:}G \xrightarrow{C,\alpha} (P' \mid Q){:}G}$	$bn(\alpha) \cap fn(Q) = \emptyset$
PROC-COM	$\dfrac{P{:}G \xrightarrow{C_1,\overline{w:G}x} P'{:}G \quad Q{:}G \xrightarrow{C_2,z:G(y)} Q'{:}G}{(P \mid Q){:}G \xrightarrow{C_1 \wedge C_2 \wedge [w=z],\tau} (P' \mid Q'\{x/y\}){:}G}$	
PROC-CLOSE	$\dfrac{P{:}G \xrightarrow{C_1,(\nu x)\overline{w:G}x} P'{:}G \quad Q{:}G \xrightarrow{C_2,z:G(x)} Q'{:}G}{(P \mid Q){:}G \xrightarrow{C_1 \wedge C_2 \wedge [w=z],\tau} ((\nu x)(P' \mid Q')){:}G}$	

Table 9: Symbolic semantics for basic ω-node expressions.

applied to each context of the expression. The symbolic semantics also permits us to introduce useful operators, such as a π-calculus-like *mismatch* operator to the ω-calculus, and provide consice semantics to such operators.

We define a symbolic semantics for the ω-calculus in a manner similar to that for the π-calculus [15]. The symbolic semantics is given in terms of a symbolic labeled transition relation. Each symbolic transition has an associated constraint representing the conditions under which that transition is enabled. More specifically, symbolic transitions are of the form $M \xrightarrow{C,\alpha} M'$, where C is a constraint on the free pnames of M. Constraints are conjunctions of zero or more atomic constraints, which are of the form **true**, **false**, and for pnames x and y, $x = y$ and $x \neq y$. An empty constraint is equivalent to **true** as are constraints of the form $x = x$. Constraints of the form $x \neq x$ are equivalent to **false**. A conjunction containing **false** is equivalent to **false**. In the following, we assume that constraints are maintained, using these equivalences, in simplified form.

The inference rules for the symbolic semantics of the ω-calculus are given in Tables 8-11. In the tables, we use λ to represent transition labels, i.e., pairs (C, α). The rules also use a constraint expression of the form $C - x$, which represents the constraint obtained from C by replacing all occurrences of $x = y$ by **false** and $x \neq y$ by **true**. We do not need to consider cases such as $x = x$ since we assume that C is in simplified form.

Rule MATCH and the rules corresponding to binary (unicast) synchronization (PROC-COM, UNI-COM, and UNI-CLOSE) generate equality constraints

Rule Name	Rule	Side Condition
STRUCT	$\dfrac{N \equiv M \quad M \xrightarrow{\lambda} M' \quad M' \equiv N'}{N \xrightarrow{\lambda} N'}$	
MOBILITY(I)	$\dfrac{}{M \mid P{:}G \xrightarrow{\mathbf{true},\mu} M \mid P{:}G'}$	$G' \neq G$, $G' \subseteq G \cup fgn(M)$, $I(M \mid P{:}G) \implies I(M \mid P{:}G')$
PAR(I)	$\dfrac{M \xrightarrow{C,\alpha} M'}{M \mid N \xrightarrow{C,\alpha} M' \mid N}$	$bn(\alpha) \cap fn(N) = \emptyset$ $I(M \mid N) \implies I(M' \mid N)$
COM	$\dfrac{M \xrightarrow{C_1,\overline{G}x} M' \quad N \xrightarrow{C_2,G'(y)} N'}{M \mid N \xrightarrow{C_1 \wedge C_2,\overline{G}x} M' \mid N'\{x/y\}}$	$G \cap G' \neq \emptyset$
GNAME-RES1	$\dfrac{M \xrightarrow{C,\alpha} M'}{(\nu\, g)M \xrightarrow{C,\alpha \setminus \{g\}} (\nu\, g)M'}$	$\alpha \in \{\tau, \mu\}$, or $gn(\alpha) \setminus \{g\} \neq \emptyset$
GNAME-RES2	$\dfrac{M \xrightarrow{C,\overline{G}x} M'}{(\nu\, g)M \xrightarrow{C,\tau} (\nu\, g)M'}$	$G = \{g\}$

Table 10: Symbolic semantics for structured ω-node expressions.

over pnames. In the non-symbolic case, we say that two nodes performing unicast send and receive operations can synchronize only if they use identical channel names. In contrast, in the symbolic case, we permit any two such operations to synchronize and generate a condition that the two channels should be the same (represented by the equality constraint between the two names). Inequality constraints are introduced by the MISMATCH rule.

Consider the PNAME-RES rule, which says that $(\nu x)M \xrightarrow{C',\alpha} (\nu x)M'$ can be inferred from $M \xrightarrow{C,\alpha} M'$ if x is not a name in α. Note that while C is a constraint on the free pnames of M, C' is a constraint on the free pnames of $(\nu x)M$; i.e., C' does not contain x. Consider an equality constraint of the form $x = y$ in C. This constraint will be unsatisfiable in the context of (νx) since x is a restricted pname. Now consider a disequality constraint of the form $x \neq y$ in C. This constraint will be always satisfiable in the context of (νx) since x is a restricted name and y is a free name. Hence we obtain C' as $C - x$: derived from C by replacing all occurrences of $x = y$ by **false** and $x \neq y$ by **true**.

The equality constraints in a conjunction of constraints induce an equivalence relation on the names appearing in the constraints. For a given constraint

C, we use σ_C to denote a substitution that maps all names in the same equivalence class to a representative name (chosen arbitrarily) of the class. We use $C_1 \triangleright C_2$ to indicate that C_1 implies C_2.

We can establish a correspondence between the symbolic semantics and the transitional semantics presented in Section 3, formalized as follows.

Theorem 12 (Correspondence). *For all M in the mismatch-free fragment of the ω-calculus: $M \xrightarrow{C,\alpha} M'$ iff $M\sigma_C \xrightarrow{\alpha\sigma_C} M'\sigma_C$.*

This theorem can be proved by induction on the derivation length.

Symbolic Bisimulation for the ω-Calculus. We now proceed to define symbolic bisimulation for the ω-calculus. As desired, the congruence results of Section 4 can be established for this extension as well.

Definition 5. *A relation $S \subseteq \mathbf{N} \times \mathbf{N}$ on nodes is a* symbolic simulation *if $M\ S\ N$ implies:*

- *$fgn(M) = fgn(N)$, and*
- *whenever $M \xrightarrow{C_1,\alpha} M'$, where $bn(\alpha)$ is fresh, there exist N', β, and C_2 s.t. $N \xrightarrow{C_2,\beta} N'$ and*
  - *$C_1 \triangleright C_2$*
  - *$\alpha\sigma_{C_1} \equiv \beta\sigma_{C_1}$*
  - *$M'\sigma_{C_1}\ S\ N'\sigma_{C_1}$.*

S is a symbolic bisimulation *if both S and S^{-1} are symbolic simulations. Nodes M and N are* symbolic bisimilar*, written $M \asymp N$, if $M\ S\ N$ for some* symbolic bisimulation *S.*

Proposition 13. *(i) $\asymp$ is an equivalence relation; and (ii) $\asymp$ is the largest symbolic bisimulation.*

Symbolic bisimulation equivalence is a congruence for the ω-calculus, as formally stated below.

Theorem 14 (Congruence for Symbolic Bisimulation for the ω-Calculus). *$\asymp$ is a congruence for the ω-calculus; i.e., for all $M_1, M_2 \in \mathbf{N}$, the following hold:*

(i) $M_1 \asymp M_2$ implies $\forall x \in \mathbf{Pn} : (\nu x)M_1 \asymp (\nu x)M_2$;
(ii) $M_1 \asymp M_2$ implies $\forall g \in \mathbf{Gn} : (\nu g)M_1 \asymp (\nu g)M_2$; and
(iii) $M_1 \asymp M_2$ implies $\forall N \in \mathbf{N} : M_1|N \asymp M_2|N$.

See Appendix B for a proof for the ω_0-calculus; proofs for the extended calculi follow along the same lines.

For mismatch-free fragment of the ω-calculus, the symbolic bisimulation and the strong (late) bisimulation coincide. The notion of weak transitions used in defining the weak bisimulation for the ω-calculus, can be lifted to the symbolic semantics to define symbolic weak transitions. A weak version of symbolic bisimulation can be defined over the symbolic weak transitions.

Comment (1k) Done. "You should explain, or at least comment, how the symbolic bisimulation relates to the strong late bisimulation. For instance, I would expect that M1 and M2 are strong late bisimilar iff M1 and M2 are symbolic bisimilar when M1 and M2 are mismatch-free?"

Comment (1l) Done. "Since your case studies involve weak bisimulation I assume you have also considered a weak symbolic bisimulation, although you don't define one, please comment and clarify."

Rule Name	**Rule**	**Side Condition**
UNI-COM	$\dfrac{M \xrightarrow{C_1,\overline{w:G}x} M' \qquad N \xrightarrow{C_2,z:G'(y)} N'}{M \mid N \xrightarrow{C_1 \wedge C_2 \wedge [w=z],\tau} M' \mid N'\{x/y\}}$	$G \cap G' \neq \emptyset$
UNI-OPEN	$\dfrac{M \xrightarrow{C,\overline{z:G}x} M'}{(\nu x)M \xrightarrow{C-x,(\nu x)\overline{z:G}x} M'}$	$x \neq z$
UNI-CLOSE	$\dfrac{M \xrightarrow{C_1,(\nu x)\overline{w:G}x} M' \qquad N \xrightarrow{C_2,z:G'(x)} N'}{M \mid N \xrightarrow{C_1 \wedge C_2 \wedge [w=z],\tau} (\nu x)(M' \mid N')}$	$G \cap G' \neq \emptyset$
OPEN	$\dfrac{M \xrightarrow{C,\overline{G}x} M'}{(\nu x)M \xrightarrow{C-x,(\nu x)\overline{G}x} M'}$	
COM-RES	$\dfrac{M \xrightarrow{C_1,(\nu x)\overline{G}x} M' \qquad N \xrightarrow{C_2,G'(x)} N'}{M \mid N \xrightarrow{C_1 \wedge C_2,(\nu x)\overline{G}x} M' \mid N'}$	$G \cap G' \neq \emptyset$
CLOSE	$\dfrac{M \xrightarrow{C,(\nu x)\overline{G}x} M'}{(\nu\, g)M \xrightarrow{C,\tau} (\nu\, g)(\nu x)M'}$	$G = \{g\}$
PNAME-RES	$\dfrac{M \xrightarrow{C,\alpha} M'}{(\nu x)M \xrightarrow{C-x,\alpha} (\nu x)M'}$	$x \notin n(\alpha)$

Table 11: Symbolic semantics for structured ω-node expressions.

6. Towards Verification of ω-Calculus Specifications

In the section, we present two developments that yield a a prototype system for the specification and verification of ω-calculus specifications. First, we extend the calculus with constructs that simplify the specification of practical MANET protocols. Secondly, we show how the transitional semantics of the ω-calculus can be directly encoded in Prolog, in order to generate the transition system corresponding to a given specification.

Syntactic extensions to the ω-calculus. The ω-calculus provides the basic mechanisms needed to model MANETs. In order to make specifications more concise, we extend the calculus to a polyadic version (along the lines of the polyadic π-calculus [11]) and add support for data types such as bounded integers and structured terms. The matching prefix is extended to include equality over these types. Terms composed of these types can be used as values in a unicast or local broadcast transmission, or as actual parameters in a process invocation. We also introduce set-valued types and permit the use of a membership operation (denoted by '$\in$') in a match. Note that finite sets can be represented by finite terms, and the test for set membership can be implemented by a sum of equality tests. Hence the addition of set-valued data can be regarded as syntactic sugar and does not affect the proofs of the properties of the calculus given in Section 4. The modifications to the theory developed in the preceding sections to account for these syntactic extensions to the calculus are straightforward.

Encoding the transitional semantics in Prolog. Following the Prolog encoding of the semantics of value-passing CCS and the π-calculus [18, 24] using the XSB tabled logic-programming system [21], we encoded the symbolic transitional semantics of the ω-calculus using Prolog rules. Each inference rule of the semantics is represented as a rule for the predicate `trans`, which evaluates the transition relation of an ω-calculus model.

In our encoding, each ω-calculus expression is represented as a Prolog term. For instance, a basic node expression of the form $P : G$ is represented by the term `basic(P, G)`, where `P` and `G` are the Prolog terms representing the process expression P and set of group names G, respectively. The key aspect of our encoding is to represent names in the ω calculus—pnames as well as gnames—as Prolog variables. This representation was used for the π-calculus in our earlier work [24].

Using this representation, several operations such as renaming of bound names need not be implemented by our encoding explicitly; the way the deductive engine of Prolog handles logical variables implements all the necessary name manipulation. For instance, our encoding of the transitional semantics does not have to handle substitutions to names explicitly (which arise in the application of process names). Moreover, it is not necessary to encode alpha-renaming; bound names are implicitly renamed when clause instances are picked by Prolog's resolution step (which renames all variables in the selected program clause). Finally, when encoding the symbolic semantics of ω-calculus, we explicitly represent only the disequality constraints on transitions (i.e., those of

```
% MCAST
trans(basic(pref(bcast(X),PExp),Gs), [], bcast(Gs,X), basic(PExp,Gs)).

% RECV
trans(basic(pref(recv(X),PExp),Gs), [], brecv(Gs,X), basic(PExp,Gs)).

% UNI-SEND
trans(basic(pref(out(Z,X),PExp),Gs), [], unisend(Z,Gs,X), basic(PExp,Gs)).

% COM
trans(par(M1, N1), C, bcast(Gs1, X), par(M2, N2)) :-
  trans(M1, C1, bcast(Gs1, X), M2),
  trans(N1, C2, brecv(Gs2, X), N2),
  non_disjoint(Gs1, Gs2), % sets Gs1 and Gs2 have common gname(s)
  and(C1, C2, C).

% GNAME-RES2
trans(nu_g(G, M1), C, tau, nu_g(G, M2)) :-
  trans(M1, C, bcast(Gs, X), M2),
  Gs == [G].

% UNI-OPEN
trans(nu(X, M1), C, unires(Z, Gs, X), M2) :-
  trans(M1, C1, unisend(Z, Gs, Y), M2),
  X == Y, Z \== X,
  remove_from_constraint(C1, X, C).  % C = C1-X
```

Figure 4: Encoding of selected symbolic transition rules in Prolog.

the form $x \neq y$); the equality constraints are processed implicitly using Prolog's unification mechanism.

In our encoding, each symbolic transition is represented by an instance of a 4-ary prolog predicate `trans`. In particular, a symbolic transition of the form $M_1 \xrightarrow{C,\alpha} M_2$ is represented by `trans(`M_1`,` C`,` α`,` M_2`)`. The derivation of a symbolic transition from the semantic rules is realized by encoding the rules as clauses defining `trans`, and using query evaluation in Prolog.

Figure 6 shows the Prolog encoding of selected symbolic transition rules of the ω-calculus. Observe that the encoding of the MCAST, RECV and UNI-SEND rules is straightforward. For these rules, the constraint **true** is represented by the empty list '`[]`'. The encoding of COM rule is also direct. Predicate `non_disjoint` is used to test for non-disjointness of two sets (represented as lists) and predicate `and` is used to compute the conjunction of two constraints. Note that in a broadcast-receive transition of the form $M \xrightarrow{C,G(y)} M'$, the name y is a bound name of M. In the symbolic semantics, we assume that before applying the COM rule, y is renamed to the name received in COM. Such alpha-renaming corresponds to an application of the STRUCT rule. In our encoding, we first ensure that all bound names are mapped to distinct Prolog variables

and are also distinct from free names. We then ensure that the receiving and sent names are identical by unifying the corresponding Prolog variables.

The GNAME-RES2 rule is applicable only when the set of group names involved in the broadcast action (`Gs` in our encoding) is a singleton set containing only `G`, the restricted group name. Since group names are encoded as variables, this check has to be performed by testing if `Gs` is *identical* to `[G]`: i.e. whether for all substitutions θ the two terms `Gs`θ and `[G]`θ are the same. This test is accomplished using the "`==`" operator in Prolog. Note that if we had used "`=`" instead, we would have incorrectly unified `Gs` with `[G]`, thereby possibly treating two distinct group names as the same.

Finally, consider the encoding of UNI-OPEN. This rule is applicable when the name sent by unicast is a restricted name. In the encoding, we apply this rule by first generating transitions from `M1`, and then checking if the name `Y` sent by unicast is same as the restricted name `X`. As in the case of GNAME-RES2, we use "`==`" to test whether two names are identical. This rule also uses "`\==`" to test whether two names are not identical (i.e. distinct).

The other symbolic transition rules of the ω-calculus are encoded similarly. As remarked earlier, the key aspect of the encoding is the representation of pnames and gnames by Prolog variables, following [24]. The soundness and completeness of our encoding can be established along the same lines as in [24].

7. Modeling and Verifying MANET Protocols using the ω-Calculus

We used our Prolog encoding of the semantics of the ω-calculus to develop and analyze formal ω-calculus models of a leader-election algorithm for MANETs [20] and the AODV routing protocol [16]. The main purpose of these case studies is to show that models of realistic MANET protocols can be constructed in the ω-calculus, and the semantics of these encodings, in terms of labeled transition systems, can be effectively computed. We use the derived transition systems to verify reachability properties of these protocols.

Added few lines

7.1. Case Study 1: A Leader Election Protocol for MANETs

The algorithm of [20] elects the node with the maximum id among a set of connected nodes as the leader of the connected component. A node that initiates a leader election sends an *election* message to its neighboring nodes. The recipients of the *election* message mark the node from which they received the message as their parent and send the *election* message to their neighbors, thereby building a spanning tree with the initiator as the root. After sending an *election* message, a node awaits acknowledgements from its children in the spanning tree. A child node n sends its parent an acknowledgement *ack* with the maximum id in the spanning tree rooted at n. The maximum id in the spanning tree is propagated up the tree to the root. The root node then announces the leader to all the nodes in its spanning tree by sending a *leader* message. To keep track of the neighbors of a node, *probe* and *reply* messages are used periodically. When a node discovers that it is disconnected from its *leader*, it initiates an election process. The flow of *election*, *ack*, and *leader* messages is depicted in Fig. 5, where the node with id 1 is the initiator.

Comment (2s) Done. "what happens if there are cycles? Please specify better"

Figure 5: Message flow in leader election protocol

Specification of the leader election protocol in the ω-calculus. We model a network as the parallel composition of basic ω-nodes, whose process interfaces reflect the initial topology of the network. Each node runs an instance of process $node(id, chan, init, elec, lid, pChan)$ defined in Fig. 6. The meaning of this process's parameters is the following: id is the node identifier; $chan$ is an input channel; $init$ indicates whether the node initiates the election process; $elec$ indicates whether the node is part of the election process; lid represents the node's knowledge of the leader id; and $pChan$ is the parent's input channel. These parameters are represented by pnames and integers.

A node may receive $election, ack$, and $leader$ messages, representing an election message, an acknowledgement to the election process, and a leader message, respectively. We need not consider $probe$ and $reply$ messages in our model because a node can broadcast to its neighbors without knowing its neighbors, and the effect of disconnection between nodes can be modeled using the choice operator. The ω-calculus model of the protocol is given in Fig. 6. The messages, their parameters, and the parameters used in the definitions appearing in Fig. 6 are explained below:

Messages: $election(sndrChan)$; $ack(maxid)$; $leader(maxid)$.

Message parameters: $sndrChan$: input channel of the sender of the message; $maxid$: maximum id seen so far by the sender of the message.

Definition parameters: id: id of the node, $chan$: input channel of the node; $init$: 1 if node initiated the election process, 0 otherwise; $elec$: 1 if node is participating in the election process, 0 otherwise; lid: node's knowledge of the leader id; $pChan$: input channel of the node's parent in the spanning tree; $sndrChan$: input channel of the sender node of the message; $maxid$: maximum id seen so far by the node.

An example specification of an eight-node network running the leader election protocol of Fig. 6 is given in Fig. 7. The initial network topology is the same as that of the network of Fig. 5. The node with id 1 ($initElection$) is designated to be the initiator of the leader-election process. The last parameter $none$ in the process invocations indicates that the parent channel is initially not known to the processes.

Pls See: added 'On some computation path in the transition system' to the property to address the comment 2(t).

Verifying the leader election protocol model. Using our implementation of the transitional semantics of the ω-calculus, we verified the following correctness property for the leader election protocol for MANETs: *On some computation path in the transition system, eventually a node with the maximum id in a connected component is elected as the leader of the component, and every node connected to it (via one or more hops) learns about it.*

Note that the reachability property stated above does not guarantee that a leader will be always computed. In fact, due to lossy communication, there will be paths in the transition system where a leader may never be elected; hence the correctness condition can be shown only using fairness assumptions, e.g.

/* A node may receive an *election* or a *leader* message. */

$node(id, chan, init, elec, lid, pChan) \stackrel{def}{=}$

$\quad \mathbf{r}(election(sndrChan)).\ processElection(id, chan, init, 1, lid, pChan, sndrChan)$

$\quad + \mathbf{r}(leader(maxid)).\ processLeader(id, chan, init, elec, lid, pChan, maxid)$

/* Node that initiates election process broadcasts *election* msg and awaits *ack* in state *awaitAck*. */

$initElection(id, chan, init, elec, lid, pChan) \stackrel{def}{=}$

$\quad \overline{\mathbf{b}}\, election(chan).\ awaitAck(id, chan, init, 1, id, none)$

/* When a node receives an *election* message it reaches the *processElection* state where it broadcasts the *election* message and goes to state *awaitAck*. */

$processElection(id, chan, init, elec, lid, pChan, sndrChan) \stackrel{def}{=}$

$\quad \overline{\mathbf{b}}\, election(chan).\ awaitAck(id, chan, init, elec, lid, sndrChan)$

/* A node in *awaitAck* state may receive an *ack* and reach *processAck* state or it may nondeterministically conclude that it has received *ack* from all its children in the spanning tree. In the latter case, it declares the leader by broadcasting a *leader* message if it is the initiator. Otherwise, it sends (unicast) an *ack* to its parent node (*pChan*) with the maximum id in the spanning tree rooted at this node. */

$awaitAck(id, chan, init, elec, lid, pChan) \stackrel{def}{=}$

$\quad chan(ack(maxid)).\ processAck(id, chan, init, elec, lid, pChan, maxid)$

$\quad + [init = 1]\, \overline{\mathbf{b}}\, leader(lid).\ node(id, chan, init, 0, lid, pChan)$

$\quad + [init = 0]\, \overline{pChan}\, ack(id, lid).\ node(id, chan, init, elec, lid, pChan)$

/* On receiving an *ack*, a node stores the maximum of the ids received in *ack* messages. */

$processAck(id, chan, init, elec, lid, pChan, maxid) \stackrel{def}{=}$

$\quad [maxid >= lid]\, awaitAck(id, chan, init, elec, maxid, pChan)$

$\quad + [maxid < lid]\, awaitAck(id, chan, init, elec, lid, pChan)$

/* On receiving a *leader* message, a node sets its *lid* parameter to the *maxid* in the *leader* message. If *maxid* is less than *lid*, then either the node was not part of the election process or did not report *ack* to its parent node (probably because it moved away from its parent). In either case, it broadcasts its *lid* as the maximum id. */

$processLeader(id, chan, init, elec, lid, pChan, sndrChan, maxid) \stackrel{def}{=}$

$\quad [maxid = lid]($

$\quad\quad [elec = 1]\ \overline{\mathbf{b}}\, leader(maxid).\ node(id, chan, init, 0, lid, pChan)$

$\quad\quad + [elec = 0]\ node(id, chan, init, 0, lid, pChan)$

$\quad)$

$\quad + [maxid > lid]\ \overline{\mathbf{b}}\, leader(maxid).\ node(id, chan, init, 0, maxid, pChan)$

$\quad + [maxid < lid]\ \overline{\mathbf{b}}\, leader(lid).\ node(id, chan, init, 0, lid, pChan)$

Figure 6: ω-calculus encoding of the leader election protocol for MANETs.

$$
\begin{aligned}
M = (\nu a)(\nu b)(\nu c)(\nu d)(\nu e)(\nu h)(\nu i)(\nu j)(\nu g_1)(\nu g_2)(\nu g_3)(\nu g_4)(\nu g_5)(\nu g_6)(\nu g_7) \\
(initElection(1, a, 1, 0, 1, none) : \{g_1, g_2\} \\
\mid node(2, b, 0, 0, 2, none) : \{g_1, g_3, g_4\} \\
\mid node(3, c, 0, 0, 3, none) : \{g_4\} \\
\mid node(4, d, 0, 0, 4, none) : \{g_2, g_5\} \\
\mid node(5, e, 0, 0, 5, none) : \{g_3\} \\
\mid node(6, h, 0, 0, 6, none) : \{g_5, g_6, g_7\} \\
\mid node(7, i, 0, 0, 7, none) : \{g_6\} \\
\mid node(8, j, 0, 0, 8, none) : \{g_7\})
\end{aligned}
$$

Figure 7: ω-calculus specification of leader election protocol for an 8-node tree-structured network.

Nodes	Tree			Ring		
	States	Transitions	Time(sec)	States	Transitions	Time(sec)
5	77	96	0.97	98	118	1.22
6	168	223	3.35	212	281	4.45
7	300	455	11.55	453	664	17.58
8	663	1073	45.85	952	1560	71.22

Table 12: Verification statistics for ω-calculus model of leader election protocol.

that message loss does not happen infinitely often. Our implementation verifies reachability properties without fairness conditions, and hence we only verify the weaker property stated above.

The verification was performed on models having *tree*- and *ring*-structured initial topologies. A distinguished node (with maximum id, for example, node 8 marked 'M' for "mobile" in Fig. 5) was free to move as long as the network remained connected. A mobility invariant was used to constrain the other nodes to remain connected to their neighbors. For verification purposes, we added a node *final* to the model that remains connected to all other nodes. A node, upon learning its leader, forwards this information to node *final*. After *final* receives messages from every other node with their leader ids equal to the maximum id in the network, it performs the observable action $action(leader(MaxId))$. The closed ω-specification of the protocol was checked for weak bisimilarity with an ω-specification that emits $action(leader(MaxId))$ as the only observable action. Weak bisimilarity between these two specifications indicates that the correctness property is true of the system.

Our Prolog encoding of the weak bisimulation checker for the ω-calculus includes the weak version of the transition relation, abstracting τ- and μ-transitions, encoded as the `dtrans` predicate. The predicate `nb(S1, S2)` checks if two ω-specifications $S1$ and $S2$ are weak bisimilar.

We verified the correctness property for networks containing 5 through 8 nodes. Table 12 lists the states, transitions and time (in seconds) it took our Prolog implementation of the calculus and weak bisimulation checker to verify

Comment 2(t) Done. "if the choice is non-deterministic, what guarantees that all acks are ever considered. Also, since nodes may not receive messages even if they are connected it is not clear to me how the property at page 29 may be true: simply the system can lose all the messages. Do you have some fairness assumptions? "

Comment (2u) Done. "Say more about the bisimilarity checker".

the property for networks with initial tree and ring topologies.

7.2. Case Study 2: The AODV Routing Protocol

The *Ad Hoc On-Demand Distance Vector* (AODV) protocol [16] is a routing protocol that discovers and maintains point-to-point routes in a MANET. Route discovery is initiated by a node on demand. If a node (source) does not know a route to a destination node to which it wants to route a packet, it initiates route discovery by locally broadcasting a *route request*. On receiving a *route request*, if a node knows a route to the destination node or is itself the destination, it responds to the sender node with a *route reply*, otherwise it forwards (locally broadcasts) the *route request*. Each route request is marked with a broadcast-id, assigned by the originator node of the request. The broadcast-id and the originator node's id uniquely define a route request, and are used to avoid processing of duplicate requests. The broadcast-id is incremented by a node everytime it originates a route request. Sequence numbers are used with route requests and route replies to maintain freshness of routes. Route error messages are used to convey invalidation of routes due to staleness of routes, indicated by a lower sequence number.

Specification of the AODV Protocol in the ω-calculus. We model a MANET as the parallel composition of basic ω-nodes. The interfaces of all nodes are initialized in accordance with the initial topology of the network. Each node in the network runs an instance of process *aodv* defined in Fig. 8. Process *aodv* has the following parameters: process identifier *id* (a pname), broadcast id *bid*, sequence number *sqn* (for messages and route requests), route table *rt* (a list of tuples), set of previously seen route requests *rreqs* (a list of tuples), and set of known destinations *kD* (a list of pnames). These parameters record the state of a node which may change as the protocol runs and the network evolves. An *aodv* process can receive a message either destined for it, or a message locally broadcasted by a neighboring node. A node may receive *data*, *rreq*, *rrep*, *rerr* messages representing data packet, route request, route reply and route invalidation, respectively. On receiving a message, the protocol may modify its state and/or broadcast a message. The *aodv* process invokes message handlers, defined using ω-process definitions, to process the received messages. Reception of *data*, *rreq*, *rrep*, and *rerr* (parameterized) messages is handled by processes defined by *pktP*, *rreqP*, *rrepP*, and *rerrP*, respectively (See Fig. 8). A route table *rt* is a set of tuples with each tuple containing id, sequence number, hop count, next hop, active neighbors, and route validity for each known destination node. Data manipulation code for updating route table (*rt* to *nrt*), extracting next hop (*y*), sequence number (*dsqn*), and active neighbors (*dactn*) for a destination, from the route table, and incrementing sequence number, broadcast id, and hop count is omitted from the encoding given in Fig. 8.

On receiving a data packet, a node accepts it if the packet is destined for it, otherwise if it knows the route to the destination, it sends the packet to the next hop towards the destination node, else it initiates a route discovery for

$aodv(id, bid, sqn, rt, rreqs, kD) \stackrel{\text{def}}{=}$
$\mathbf{r}(msg).\ ([msg = pkt(data, did, sndrid)]$
$pktP(data, did, sndrid, id, bid, sqn, rt, rreqs, kD)$
$+\ [msg = rreq(hops, mbid, did, dsqn, srcid, ssqn, sndrid)]$
$rreqP(hops, mbid, did, dsqn, srcid, ssqn, sndrid, id, bid, sqn, rt, rreqs, kD)$
$+\ [msg = rrep(hops, did, dsqn, srcid, sndrid)]$
$rrepP(hops, did, dsqn, srcid, sndrid, id, bid, sqn, rt, rreqs, kD)$
$+\ [msg = rerr(did, dsqn, sndrid)]$
$rerrP(did, dsqn, sndrid, id, bid, sqn, rt, rreqs, kD)\ )$

$pktP(data, did, sndrid, id, bid, sqn, rt, rreqs, kD) \stackrel{\text{def}}{=}$
$[did = id]\ aodv(id, bid, sqn, rt, rreqs, kD)$
$+\ [did \neq id]$ /* y is the next hop node towards did. nrt is obtained by adding $sndrid$ to $actn$ of did in rt */
$(\ [did \in kD]\ \ \overline{y}\, pkt(data, did, id).\ aodv(id, bid, sqn, nrt, rreqs, kD)$
$+$ /* $newbid$ is $bid + 1$ and $rdsqn$ is the sequence number for did in rt */
$[did \notin kD]\ \ \overline{\mathbf{b}}\, rreq(0, newbid, did, rdsqn, id, sqn, id).$
$aodv(id, newbid, sqn, rt, rreqs, kD)\ )$

$rreqP(hops, mbid, did, dsqn, srcid, ssqn, sndrid, id, bid, sqn, rt, rreqs, kD) \stackrel{\text{def}}{=}$
$[(srcid, mbid) \in rreqs]\ aodv(id, bid, sqn, rt, rreqs, kD)$
$+\ [(srcid, mbid) \notin rreqs]\ ($
/* y is the next hop node towards $srcid$. maxsqn is the maximum of sqn and $dsqn$. nrt is obtained by updating the route to $srcid$ in rt. */
$[did = id]\ \ \overline{\mathbf{b}}\, rrep(y, 0, id, maxsqn, srcid, id).$
$aodv(id, bid, maxsqn, nrt, rreqs, kD)$
$+$ /* $dhops$ is the number of hops towards did in rt. $rsqn$ is the sequence number for did. $nhops$ is $hops + 1$. */
$[did \neq id]\ \ (\ [did \in kD]\ \ \overline{\mathbf{b}}\, rrep(y, dhops, did, rsqn, srcid, id).$
$aodv(id, bid, sqn, nrt, rreqs, kD)$
$+\ [did \notin kD]\ \ \overline{\mathbf{b}}\, rreq(nhops, mbid, did, dsqn, srcid, ssqn).$
$aodv(id, bid, sqn, nrt, rreqs, kD)\ \ )\ )$

$rrepP(hops, did, dsqn, srcid, sndrid, id, bid, sqn, rt, rreqs, kD) \stackrel{\text{def}}{=}$
/* nrt is obtained by updating the route to did in rt. */
$[srcid = id]\ \ aodv(id, bid, sqn, nrt, rreqs, kD)$
$+$ /* y is the next hop node towards $srcid$. $nhops$ is $hops + 1$. */
$[srcid \neq id]\ \ \overline{y}\, rrep(nhops, did, dsqn, srcid, id).\ \ aodv(id, bid, sqn, nrt, rreqs, kD)$

$rerrP(did, dsqn, sndrid, id, bid, sqn, rt, rreqs, kD) \stackrel{\text{def}}{=}$
$[did \in kD]$ /* $dactn$ are active neighbors for did in rt.
nrt is obtained by invalidating the route to did. */
$notifyAllRErr(dactn, rerr(did, dqsn, id), id, bid, sqn, nrt, rreqs, kD)$
$+\ [did \notin kD]\ \ aodv(id, bid, sqn, rt, rreqs, kD)$

$notifyAllRErr(actn, msg, id, bid, sqn, rt, rreqs, kD) \stackrel{\text{def}}{=}$
$[actn = []]\ \ aodv(id, bid, sqn, rt, rreqs, kD)$
$+\ [actn \neq []]\ \ notifyRErr(actn, msg, id, bid, sqn, rt, rreqs, kD)$

$notifyRErr(actn, msg, id, bid, sqn, rt, rreqs, kD) \stackrel{\text{def}}{=}$
/* x is an element in $actn$ and $remactn$ are remaining elements of $actn$ */
$\overline{x}\, msg.\ notifyAllRErr(remactn, msg, id, bid, sqn, rt, rreqs, kD)$

Figure 8: Encoding of the AODV protocol in the ω-calculus.

the destination node. On receiving a route request *rreq*, a node replies with *rrep*, if it knows a route to the destination, otherwise it forwards the *rreq* via local broadcast. Each such request is associated with a broadcast id (*mbid*) set by the originator (identified by *srcid*) of the message. A route request *rreq* is discarded if it had been received previously ($(srcid, mbid) \in rreqs$). Otherwise, the route table is updated (to *nrt*) with a route to node *srcid*. If the node itself is the destination node (identified by *did*) to which the route is sought, or if the node knows a route to the destination ($did \in kD$), a route-reply message (*rrep*) is sent. Otherwise, the node locally broadcasts the *rreq* message (via action $\overline{\mathbf{b}}$) with the hop count (*hops*) incremented by one (to *nhops*). On receiving a route reply *rrep*, a node updates its route table accordingly. If the node itself is not the initiator of the corresponding *rreq*, it forwards the *rrep* to the next hop towards the initiator node. Detection of a change in network topology is modeled using non-determinism. On detection of a change in network topology, a node invalidates the route table entry for the disconnected neighbor node, and sends a route error *rerr* to the affected nodes.

Verifying the AODV protocol model. Using our Prolog encoding of the transitional semantics of the ω-calculus, we verified a simplified version of the AODV routing protocol. The simplified version ignores sequence numbers and uses two distinguished nodes as the source and destination nodes for the route discovery process. Broadcast id (*bid*) and hop count (*hops*) are modeled as bounded integers. Routes get invalidated due to node movement or link failures along the route. We verified following properties:

- *deadlock-freedom:* There is no state in the model without an outgoing transition.
- *route-found:* As long as there exists a path from a source node to a destination node during route detection, on some computation path in the transition system it will eventually be detected.

It should be noted that, similar to the leader-election property, the *route-found* property is a weaker form of the correctness condition that a route is always found provided node mobility and message loss do not occur infinitely often.

The verification was performed on models with initial *line* topologies, with the destination node being 1-, 2-, 3-hops away from the source node in networks with 2, 3, and 4 nodes, respectively. The network topology was allowed to change freely during verification. The *deadlock-freedom* property involved searching for states with no transitions. In the model, when a node has found a route it performs an external action *action(routeFound)*. The *route-found* property was verified by checking for reachability of a transition labeled *action(routeFound)* from the start state of the model. Table 13 lists the number of states and transitions generated using our XSB-based implementation of the ω-calculus for network models containing 2, 3 and 4 nodes, as well as the time (in seconds) it took to verify deadlock-freedom and route-found properties.

Pls see: Modified route-found to a reachability property to address comments 1(m) and 2(v).

Comment 1(m) Done. Why does a transition labeled with the action routeFound guarantee a route will be eventually found, it sounds more that there is a possibility to find a route, which is a much weaker requirment? Clarify.

Comment 2(v) Done. 'route-found': "how can this be possible, if messages may be lost?"

Nodes	Deadlock Freedom			Route Found		
	States	Transitions	Time(sec)	States	Transitions	Time(sec)
2	8	16	0.07	5	10	0.06
3	30	78	0.25	15	39	0.16
4	380	1440	4.56	191	732	2.74

Table 13: Verification statistics for ω-calculus model of AODV protocol.

7.3. Discussion

We consider our current implementation of the calculus to be a prototype. Its main purpose is to demonstrate the feasibility and straightforwardness of implementing the calculus in a tabled logic-programming system. As future work, we plan to develop an optimizing compiler for the ω-calculus, along the lines of one for the π-calculus implemented in the MMC model checker [23]. As these prior results demonstrate, this should significantly improve the performance of our implementation.

We observed a number of benefits in using the ω-calculus to model the leader election protocol for MANETs and the AODV routing protocol. (1) Neither of these protocols assumes reliable communication. This fits well with the ω-calculus semantics which models lossy broadcast. (2) The concise and modular nature of our specifications is a direct consequence of the calculus's basic features, including separation of control behavior (processes) from neighborhood information (interfaces), and modeling support for unicast, local broadcast, and mobility. (3) The mobility constraints imposed on the leader election protocol model (Section 7.1) are specified independently of the control logic using a mobility invariant. For the case at hand, the invariant dictates that all nodes other than a distinguished node (node 8 in Fig. 5) remain connected to their initial neighbors. Thus, during protocol execution, process interfaces may change at will as long as the mobility invariant is maintained. (4) Our specifications of the leader-election protocol and the AODV protocol are given in the finite-control sub-calculus of the ω-calculus, thereby rendering them amenable to automatic verification; see also Theorem 2.

Comment (2w) Done. Changed 'direction' to 'direct'.

8. Related Work

Several process calculi have recently been developed for wireless and mobile ad hoc networks. The closest to our work are CBS# [13], CWS [10], CMN [9], and CMAN [6]. These calculi provide local broadcast and separate control behavior from neighborhood information. However, there are significant differences between these calculi and ours, which we now discuss. CBS# [13], based on the CBS process algebra of [17], supports a notion of located processes. Node connectivity information is given independently of a system specification in terms of node connectivity graphs. The effect of mobility is achieved by nondeterministically choosing a node connectivity graph from a family of such graphs when a transition is derived. In contrast, the ω-calculus offers a single, integrated language for specifying control behavior and connectivity information, and permits

reasoning about changes to connectivity information within the calculus itself.

In CWS [10], node location and transmission range are a part of the node syntax. Node movement is not supported, although the authors suggest the addition of primitives for this feature. CWS is well-suited for modeling device-level behaviors (e.g., interference due to simultaneous transmissions) in wireless systems.

In CMN [9], a MANET node is a named, located sequential process that can broadcast within a specific transmission radius. Both the location and transmission radius are values in a physical coordinate system. Nodes are designated as mobile or stationary, and those of the former kind can move to an arbitrary location (resulting in a tau-transition). Bisimulation as defined for CMN is based on a notion of physically located observers. A calculus based on physical locations may pose problems for model checking as a model's state space would be infinite if locations are drawn from a real coordinate system.

In CMAN [6], each node is associated with a specific *location*. Furthermore, each node n is annotated by a *connection set*: the set of locations of nodes to which n is connected. Connections sets thus determine the network topology. Synchronous local broadcast is the sole communication primitive. The connection set of a node explicitly identifies the node's neighbors. Consequently, when a node moves, its neighbors actively participate by removing from (or adding to) their connection sets the location of the moving node. This explicit handling of connection information affects the modularity of the calculus's semantics (the definition of bisimulation, in particular), and may preclude reasoning about open systems. In contrast, in the ω-calculus, neighborhood information is implicitly maintained using groups, thereby permitting us to define bisimulation relations in a natural way.

Other calculi for mobile processes that have been proposed in the literature include the π-calculus [12], HOBS [14], distributed process calculus $D\pi$ [8], and the ambient calculus [3]. These calculi do not support primitive for broadcast. Some calculi that support broadcast as a primitive are the $b\pi$-calculus [5] and PRISMA [2]. The $b\pi$-calculus adds broadcast communication as a primitive to the π-calculus and provides same mechanism for mobility as the π-calculus. PRISMA is a parametric calculus that can be instantiated with different interaction policies, and provides a uniform framework for expressing different synchronization models such as unicast and broadcast. Mobility in PRISMA is provided by name-passing as in the π-calculus. These calculi could be used to model MANETs but not as in a concise and natural fashion as with the ω-calculus because they intermix specification of network structure with the specification of the control behavior of a protocol.

Comment (2x) Done. Add about PRISMA TCS 2008 paper. Also included discussion about bpi-calculus in this para.

9. Conclusions and Future Work

The ω-calculus, introduced in this paper, is a conservative extension of the π-calculus that permits succinct and high-level encodings of MANET systems and protocols. The salient aspect of the calculus is its group-based support for local broadcast communication over dynamically changing network topologies. We have shown that reachability of system states is decidable for the finite-

control fragment of the calculus, and late bisimulation equivalence and its weak counterpart are a congruence.

Commented the symbolic semantics part in the conclusion.

We illustrated the practical utility of the ω-calculus by using it to develop models of a leader-election algorithm for MANETS [20] and the AODV routing protocol [16]. We also showed how the calculus's operational semantics can be readily encoded in the XSB tabled logic-programming system, thereby permitting the generation of transition systems from ω-calculus specifications. We used this feature to implement a weak bisimulation checker for the ω-calculus, which we then used to verify certain key properties of our encodings of the leader election algorithm of [20] and the AODV routing protocol [16].

As mentioned in Section 7, future work involves the development of an optimizing compiler for the ω-calculus, along the lines of one for the π-calculus implemented in the MMC model checker [23]. MMC exploits the use of binary synchronization in the π-calculus, generating specialized rules from which the transition system can be derived efficiently at model-checking time. The MMC compiler enables MMC to match the efficiency of model checkers for non-mobile systems. Extending such compilation techniques to broadcast and multicast communication is an open problem. Another avenue of future work is the development of a compositional model checker for the ω-calculus, such as those for CCS and the π-calculus [1, 22]. A model checker of this nature would permit verification of infinite families of MANETs. Finally, the ω-calculus models bidirectional connectivity between nodes. Since certain MANET protocols rely on unidirectional node connections, it would be fruitful to extend the calculus with such a modeling capability.

Comment 2(y) Done. Changed 'such as of those' to 'such as those'.

Acknowledgements. We would like to thank the anonymous reviewers for their valuable comments which substantially helped to improve the quality of the paper. Research supported in part by NSF grants CCR-0205376, CNS-0509230, CNS-0627447, and ONR grant N000140710928.

References

[1] S. Basu and C. R. Ramakrishnan. Compositional analysis for verification of parameterized systems. In *TACAS*, volume 2619 of *LNCS*, pages 315–330. Springer, 2003.

[2] R. Bruni and I. Lanese. Parametric synchronizations in mobile nominal calculi. *Theor. Comput. Sci.*, 402(2-3):102–119, 2008.

[3] L. Cardelli and A. D. Gordon. Mobile ambients. In *FOSSACS*. Springer-Verlag, 1998.

[4] M. Dam. On the decidability of process equivalences for the π-calculus. *Theoretical Computer Science*, 183:215–228, 1997.

[5] C. Ene and T. Muntean. A broadcast-based calculus for communicating systems. In *IPDPS '01: Proceedings of the 15th International Parallel & Distributed Processing Symposium*, page 149, Washington, DC, USA, 2001. IEEE Computer Society.

[6] J. C. Godskesen. A calculus for mobile ad hoc networks. In *COORDINATION*, volume 4467 of *Lecture Notes in Computer Science*, pages 132–150. Springer, 2007.

[7] J. C. Godskesen and O. Gryn. Modelling and verification of security protocols for ad hoc networks using UPPAAL. In *Proc. 18th Nordic Workshop on Programming Theory*, Oct. 2006.

[8] M. Hennessy and J. Riely. Resource access control in systems of mobile agents. In *High-Level Concurrent Languages*, volume 16.3 of *Electr. Notes Theor. Comput. Sci.*, pages 3–17, 1998.

[9] M. Merro. An observational theory for mobile ad hoc networks. In *Proc. MFPS '07*, volume 173 of *Electr. Notes Theor. Comput. Sci.*, pages 275–293. Elsevier, 2007.

[10] N. Mezzetti and D. Sangiorgi. Towards a calculus for wireless systems. In *Proc. MFPS '06*, volume 158 of *Electr. Notes Theor. Comput. Sci.*, pages 331–354. Elsevier, 2006.

[11] R. Milner. The polyadic pi-calculus: a tutorial. In *Logic and Algebra of Specification*, pages 203–246. Springer-Verlag, 1993.

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

[13] S. Nanz and C. Hankin. A framework for security analysis of mobile wireless networks. *Theoretical Computer Science*, 367(1-2):203–227, 2006.

[14] K. Ostrovsky, K. V. S. Prasad, and W. Taha. Towards a primitive higher order calculus of broadcasting systems. In *PPDP*, pages 2–13. ACM, 2002.

[15] J. Parrow. An introduction to the pi-calculus. In Bergstra, Ponse, and Smolka, editors, *Handbook of Process Algebra*, pages 479–543. Elsevier, 2001.

[16] C. E. Perkins, E. M. Belding-Royer, and S. R. Das. Ad hoc on-demand distance vector routing protocol. Internet-draft, IETF MANET Working Group, 2003. Available at `http://www.ietf.org/internet-drafts/draft-ietf-manet-aodv-13.txt`.

[17] K. V. S. Prasad. A calculus of broadcasting systems. *Sci. Comput. Program.*, 25(2-3):285–327, 1995.

[18] Y. S. Ramakrishna, C. R. Ramakrishnan, I. V. Ramakrishnan, S. A. Smolka, T. Swift, and D. S. Warren. Efficient model checking using tabled resolution. In *CAV*, volume 1254 of *LNCS*, pages 143–154. Springer, 1997.

[19] D. Sangiorgi. On the bisimulation proof method. *Mathematical. Structures in Comp. Sci.*, 8(5):447–479, 1998.

[20] S. Vasudevan, J. F. Kurose, and D. F. Towsley. Design and analysis of a leader election algorithm for mobile ad hoc networks. In *ICNP*, pages 350–360. IEEE Computer Society, 2004.

[21] XSB. The XSB logic programming system. `http://xsb.sourceforge.net`.

[22] P. Yang, S. Basu, and C. R. Ramakrishnan. Parameterized verification of pi-calculus systems. In *TACAS*, volume 3920 of *LNCS*, pages 42–57. Springer, 2006.

[23] P. Yang, Y. Dong, C. R. Ramakrishnan, and S. A. Smolka. A provably correct compiler for efficient model checking of mobile processes. In *PADL*, volume 3350 of *LNCS*, pages 113–127. Springer, 2005.

[24] P. Yang, C. R. Ramakrishnan, and S. A. Smolka. A logical encoding of the pi-calculus: Model checking mobile processes using tabled resolution. *International Journal on Software Tools for Technology Transfer (STTT)*, 6(1):38–66, 2004.

Appendix

A. Proof of Lemma 8

Lemma 8. For all nodes $M_1, M_2 \in \mathbf{N_{nf}}$, i.e., M_1, M_2 are in normal form, the following hold:

(i) $M_1 \sim M_2$ implies $\forall x \in \mathbf{Pn} : (\nu x)M_1 \sim (\nu x)M_2$;
(ii) $M_1 \sim M_2$ implies $\forall g \in \mathbf{Gn} : (\nu g)M_1 \sim (\nu g)M_2$; and
(iii) $M_1 \sim M_2$ implies $\forall N \in \mathbf{N_{nf}}$: $M_1|N \sim M_2|N$.

Proof. We show parts *(i–iii)* of the lemma simultaneously by considering the set $S = \{((\nu\tilde{g})(\nu\tilde{x})(M_1|N), (\nu\tilde{g})(\nu\tilde{x})(M_2|N)) \mid M_1 \sim M_2, \tilde{g} \subseteq \mathbf{Gn}, \tilde{x} \subseteq \mathbf{Pn}, M_1, M_2, N \in \mathbf{N_{nf}}\}$. Following Lemma 4 it is sufficient to show that S is a strong bisimulation upto $\equiv$ to establish this lemma.

Note that if $M_1 \sim M_2$ then $fgn(M_1) = fgn(M_2)$, and hence $fgn((\nu\tilde{g})(\nu\tilde{x})(M_1|N)) = fgn((\nu\tilde{g})(\nu\tilde{x})(M_2|N))$ for all $\tilde{g}, \tilde{x}$ and N. We then show that every transition from $(\nu\tilde{g})(\nu\tilde{x})(M_1|N)$ can be matched by $(\nu\tilde{g})(\nu\tilde{x})(M_2|N)$ by considering the derivations of transitions. Transitions for $(\nu\tilde{g})(\nu\tilde{x})(M_1|N)$ can be derived by use of rules CLOSE, GNAME-RES1, GNAME-RES2, MOBILITY, PAR, UNI-COM, UNI-CLOSE, COM, COM-RES, UNI-OPEN, OPEN and PNAME-RES. Only the last three steps of each transition derivation are considered in the proof. Most importantly, following Lemma 7, we do not need to consider derivations that use STRUCT rules in the last two steps. From the structural operational semantics, the last step of a derivation will be due to the outermost $(\nu\tilde{g})$ in the expression, the next-to-last step due to the $(\nu\tilde{x})$ following the outermost $(\nu\tilde{g})$, and the earliest of the three steps due to the parallel composition $(M_1|N)$.

We omit in the proof the symmetric cases arising due to the commutativity of the parallel operator '|'. This gives rise to 15 cases (combinations of rules in the last three steps in a derivation).

1. Case CLOSE, OPEN, COM: $(\nu\tilde{g})(\nu\tilde{x})(M_1|N) \xrightarrow{\tau} (\nu\tilde{g})(\nu\tilde{x})(M_1'|N'\{x'/y\})$ given $M_1 \xrightarrow{\overline{G}x'} M_1'$ and $N \xrightarrow{G'(y)} N'$. The derivation is as follows, where $\tilde{x}_1 = \tilde{x} \setminus \{x'\}$.

$$\text{COM:}\ \frac{M_1 \xrightarrow{\overline{G}x'} M_1' \quad N \xrightarrow{G'(y)} N'}{M_1|N \xrightarrow{\overline{G}x'} M_1'|N'\{x'/y\}}\ G \cap G' \neq \emptyset$$

$$\text{OPEN:}\ \frac{}{(\nu\tilde{x})(M_1|N) \xrightarrow{(\nu x')\overline{G}x'} (\nu\tilde{x}_1)(M_1'|N'\{x'/y\})}$$

$$\text{CLOSE:}\ \frac{}{(\nu\tilde{g})(\nu\tilde{x})(M_1|N) \xrightarrow{\tau} (\nu\tilde{g})(\nu x')(\nu\tilde{x}_1)(M_1'|N'\{x'/y\})}\ G \setminus \tilde{g} = \emptyset$$

Since $M_1 \sim M_2$, $M_1 \xrightarrow{\overline{G}x'} M_1'$ means that there is an M_2' such that $M_2 \xrightarrow{\overline{G}x'} M_2'$ and $M_1' \sim M_2'$. Moreover, there exist expressions M_{N1}', M_{N2}' and N_N' in normal form such that $M_1' \equiv M_{N1}'$, $M_2' \equiv M_{N2}'$ and $N'\{x'/y\} \equiv N_N'$. Now, since $M_1' \sim M_2'$, we know $M_{N1}' \sim M_{N2}'$. Hence by construction of S, we can conclude that the pair

$(\ (\nu\tilde{g})(\nu x')(\nu\tilde{x}_1)(M'_{N1}|N'_N), (\nu\tilde{g})(\nu x')(\nu\tilde{x}_1)(M'_{N2}|N'_N)\ ) \in S$, and hence $((\nu\tilde{g})(\nu\tilde{x})(M'_1|N'\{x'/y\}), (\nu\tilde{g})(\nu\tilde{x})(M'_2|N'\{x'/y\})) \in \equiv S \equiv$.

2. Case CLOSE, OPEN, PAR:

   $(\nu\tilde{g})(\nu\tilde{x})(M_1|N) \xrightarrow{\tau} (\nu\tilde{g})(\nu\tilde{x})(M'_1|N)$ given $M_1 \xrightarrow{\overline{G}x'} M'_1$, where $\tilde{x}_1 = \tilde{x} \setminus \{x'\}$. The derivation is given below:

$$\text{PAR:}\ \frac{M_1 \xrightarrow{\overline{G}x'} M'_1}{M_1|N \xrightarrow{\overline{G}x'} M'_1|N}$$

$$\text{OPEN:}\ \frac{M_1|N \xrightarrow{\overline{G}x'} M'_1|N}{(\nu\tilde{x})(M_1|N) \xrightarrow{(\nu x')\overline{G}x'} (\nu\tilde{x_1})(M'_1|N)}$$

$$\text{CLOSE:}\ \frac{(\nu\tilde{x})(M_1|N) \xrightarrow{(\nu x')\overline{G}x'} (\nu\tilde{x_1})(M'_1|N)}{(\nu\tilde{g})(\nu\tilde{x})(M_1|N) \xrightarrow{\tau} (\nu\tilde{g})(\nu x')(\nu\tilde{x}_1)(M'_1|N)} \quad G \setminus \tilde{g} = \emptyset$$

   Since $M_1 \sim M_2$, $M_1 \xrightarrow{\overline{G}x'} M'_1$ means that there is an M'_2 such that $M_2 \xrightarrow{\overline{G}x'} M'_2$ and $M'_1 \sim M'_2$. Moreover, there exist expressions M'_{N1} and M'_{N2} in normal form such that $M'_1 \equiv M'_{N1}$ and $M'_2 \equiv M'_{N2}$. Now, since $M'_1 \sim M'_2$, we know $M'_{N1} \sim M'_{N2}$. Hence by construction of S, we can conclude that the pair $((\nu\tilde{g})(\nu\tilde{x})(M'_{N_1}|N), (\nu\tilde{g})(\nu\tilde{x})(M'_{N_2}|N)) \in S$, and hence $((\nu\tilde{g})(\nu\tilde{x})(M'_1|N), (\nu\tilde{g})(\nu\tilde{x})(M'_2|N)) \in \equiv S \equiv$.

3. Case CLOSE, PNAME-RES, COM-RES:

   $(\nu\tilde{g})(\nu\tilde{x})(M_1|N) \xrightarrow{\tau} (\nu\tilde{g})(\nu\tilde{x}_1)(M'_1|N')$ given $M_1 \xrightarrow{(\nu x')\overline{G}x'} M'_1$ and $N \xrightarrow{G'(x')} N'$ where $\tilde{x}_1 = \tilde{x} \cup \{x'\}$. The derivation is given below:

$$\text{COM-RES:}\ \frac{M_1 \xrightarrow{(\nu x')\overline{G}x'} M'_1 \quad N \xrightarrow{G'(x')} N'}{M_1|N \xrightarrow{(\nu x')\overline{G}x'} M'_1|N'} \quad G \cap G' \neq \emptyset$$

$$\text{PNAME-RES:}\ \frac{M_1|N \xrightarrow{(\nu x')\overline{G}x'} M'_1|N'}{(\nu\tilde{x})(M_1|N) \xrightarrow{(\nu x')\overline{G}x'} (\nu\tilde{x})(M'_1|N')} \quad x' \notin \tilde{x}$$

$$\text{CLOSE:}\ \frac{(\nu\tilde{x})(M_1|N) \xrightarrow{(\nu x')\overline{G}x'} (\nu\tilde{x})(M'_1|N')}{(\nu\tilde{g})(\nu\tilde{x})(M_1|N) \xrightarrow{\tau} (\nu\tilde{g})(\nu x')(\nu\tilde{x})(M'_1|N')} \quad G \setminus \tilde{g} = \emptyset$$

   Since $M_1 \sim M_2$, $M_1 \xrightarrow{(\nu x')\overline{G}x'} M'_1$ means that there is an M'_2 such that $M_2 \xrightarrow{(\nu x')\overline{G}x'} M'_2$ and $M'_1 \sim M'_2$. Moreover, there exist expressions M'_{N1}, M'_{N2} and N'_N in normal form such that $M'_1 \equiv M'_{N1}$, $M'_2 \equiv M'_{N2}$ and $N' \equiv N'_N$. Now, since $M'_1 \sim M'_2$, we know $M'_{N_1} \sim M'_{N_2}$. Hence by construction of S, we can conclude that the pair $((\nu\tilde{g})(\nu\tilde{x_1})(M'_{N_1}|N'_N), (\nu\tilde{g})(\nu\tilde{x_1})(M'_{N_2}|N'_N)) \in S$, and hence $((\nu\tilde{g})(\nu\tilde{x_1})(M'_1|N'), (\nu\tilde{g})(\nu\tilde{x_1})(M'_2|N')) \in \equiv S \equiv$.

4. Case CLOSE, PNAME-RES, PAR:

   $(\nu\tilde{g})(\nu\tilde{x})(M_1|N) \xrightarrow{\tau} (\nu\tilde{g})(\nu\tilde{x}_1)(M'_1|N)$ given $M_1 \xrightarrow{(\nu x')\overline{G}x'} M'_1$, where $\tilde{x}_1 = \tilde{x} \cup \{x'\}$. The derivation is given below:

$$\text{PAR:}\quad \frac{M_1 \overset{(\nu x')\overline{G}x'}{\longrightarrow} M_1'}{M_1|N \overset{(\nu x')\overline{G}x'}{\longrightarrow} M_1'|N} \quad x' \cap fn(N) = \emptyset$$

$$\text{PNAME-RES:}\quad \frac{M_1|N \overset{(\nu x')\overline{G}x'}{\longrightarrow} M_1'|N}{(\nu\tilde{x})(M_1|N) \overset{(\nu x')\overline{G}x'}{\longrightarrow} (\nu\tilde{x})(M_1'|N)} \quad x' \notin \tilde{x}$$

$$\text{CLOSE:}\quad \frac{(\nu\tilde{x})(M_1|N) \overset{(\nu x')\overline{G}x'}{\longrightarrow} (\nu\tilde{x})(M_1'|N)}{(\nu\tilde{g})(\nu\tilde{x})(M_1|N) \overset{\tau}{\longrightarrow} (\nu\tilde{g})(\nu x')(\nu\tilde{x})(M_1'|N)} \quad G \setminus \tilde{g} = \emptyset$$

Since $M_1 \sim M_2$, $M_1 \overset{(\nu x')\overline{G}x'}{\longrightarrow} M_1'$ means that there is an M_2' such that $M_2 \overset{(\nu x')\overline{G}x'}{\longrightarrow} M_2'$ and $M_1' \sim M_2'$. Moreover, there exist expressions M'_{N_1} and M'_{N_2} in normal form such that $M_1' \equiv M'_{N_1}$ and $M_2' \equiv M'_{N_2}$. Now, since $M_1' \sim M_2'$, we know $M'_{N_1} \sim M'_{N_2}$. Hence by construction of S, we can conclude that the pair $((\nu\tilde{g})(\nu\tilde{x_1})(M'_{N_1}|N), (\nu\tilde{g})(\nu\tilde{x_1})(M'_{N_2}|N)) \in S$, and hence $((\nu\tilde{g})(\nu\tilde{x_1})(M_1'|N), (\nu\tilde{g})(\nu\tilde{x_1})(M_2'|N)) \in \equiv S \equiv$.

5. Case GNAME-RES1, UNI-OPEN, PAR:

   $(\nu\tilde{g})(\nu\tilde{x})(M_1|N) \overset{(\nu x')\overline{z:G''}x'}{\longrightarrow} (\nu\tilde{g})(\nu\tilde{x}_1)(M_1'|N)$ given $M_1 \overset{\overline{z:G}x'}{\longrightarrow} M_1'$, where $\tilde{x_1} = \tilde{x} \setminus \{x'\}$ and $G'' = G \setminus \tilde{g}$. The derivation is given below:

$$\text{PAR:}\quad \frac{M_1 \overset{\overline{z:G}x'}{\longrightarrow} M_1'}{M_1|N \overset{\overline{z:G}x'}{\longrightarrow} M_1'|N}$$

$$\text{UNI-OPEN:}\quad \frac{M_1|N \overset{\overline{z:G}x'}{\longrightarrow} M_1'|N}{(\nu\tilde{x})(M_1|N) \overset{(\nu x')\overline{z:G}x'}{\longrightarrow} (\nu\tilde{x_1})(M_1'|N)} \quad x' \neq z, z \notin \tilde{x}$$

$$\text{GNAME-RES1:}\quad \frac{(\nu\tilde{x})(M_1|N) \overset{(\nu x')\overline{z:G}x'}{\longrightarrow} (\nu\tilde{x_1})(M_1'|N)}{(\nu\tilde{g})(\nu\tilde{x})(M_1|N) \overset{(\nu x')\overline{z:G''}x'}{\longrightarrow} (\nu\tilde{g})(\nu\tilde{x_1})(M_1'|N)} \quad G'' \neq \emptyset$$

Since $M_1 \sim M_2$, $M_1 \overset{\overline{z:G}x'}{\longrightarrow} M_1'$ means that there is an M_2' such that $M_2 \overset{\overline{z:G}x'}{\longrightarrow} M_2'$ and $M_1' \sim M_2'$. Moreover, there exist expressions M'_{N_1} and M'_{N_2} in normal form such that $M_1' \equiv M'_{N_1}$ and $M_2' \equiv M'_{N_2}$. Now, since $M_1' \sim M_2'$, we know $M'_{N_1} \sim M'_{N_2}$. Hence, by construction of S, we can conclude that the pair $((\nu\tilde{g})(\nu\tilde{x_1})(M'_{N_1}|N), (\nu\tilde{g})(\nu\tilde{x_1})(M'_{N_2}|N)) \in S$, and hence $((\nu\tilde{g})(\nu\tilde{x_1})(M_1'|N), (\nu\tilde{g})(\nu\tilde{x_1})(M_2'|N)) \in \equiv S \equiv$.

6. Case GNAME-RES1, OPEN, COM:

   $(\nu\tilde{g})(\nu\tilde{x})(M_1|N) \overset{(\nu x')\overline{G''}x'}{\longrightarrow} (\nu\tilde{g})(\nu\tilde{x}_1)(M_1'|N'\{x'/y\})$ given $M_1 \overset{\overline{G}x'}{\longrightarrow} M_1'$ and $N \overset{G'(y)}{\longrightarrow} N'$, where $\tilde{x_1} = \tilde{x} \setminus \{x'\}$ and $G'' = G \setminus \tilde{g}$. The derivation is given below:

$$\text{COM: } \frac{M_1 \xrightarrow{\overline{G}x'} M_1' \quad N \xrightarrow{G'(y)} N'}{M_1|N \xrightarrow{\overline{G}x'} M_1'|N'\{x'/y\}} \; G \cap G' \neq \emptyset$$

$$\text{OPEN: } \frac{}{(\nu\tilde{x})(M_1|N) \xrightarrow{(\nu x')\overline{G}x'} (\nu\tilde{x_1})(M_1'|N'\{x'/y\})}$$

$$\text{GNAME-RES1: } \frac{}{(\nu\tilde{g})(\nu\tilde{x})(M_1|N) \xrightarrow{(\nu x')\overline{G''}x'} (\nu\tilde{g})(\nu\tilde{x_1})(M_1'|N'\{x'/y\})} \; G'' \neq \emptyset$$

Since $M_1 \sim M_2$, $M_1 \xrightarrow{\overline{G}x'} M_1'$ means that there is an M_2' such that $M_2 \xrightarrow{\overline{G}x'} M_2'$ and $M_1' \sim M_2'$. Moreover, there exist expressions M'_{N_1}, M'_{N_2} and N'_N in normal form such that $M_1' \equiv M'_{N_1}$, $M_2' \equiv M'_{N_2}$ and $N'\{x'/y\} \equiv N'_N$. Now, since $M_1' \sim M_2'$, we know $M'_{N_1} \sim M'_{N_2}$. Hence, by construction of S, we can conclude that the pair $((\nu\tilde{g})(\nu\tilde{x_1})(M'_{N_1}|N'_N), (\nu\tilde{g})(\nu\tilde{x_1})(M'_{N_2}|N'_N)) \in S$, and hence $((\nu\tilde{g})(\nu\tilde{x_1})(M_1'|N'\{x'/y\}), (\nu\tilde{g})(\nu\tilde{x_1})(M_2'|N'\{x'/y\})) \in \equiv S \equiv$.

7. Case GNAME-RES1, OPEN, PAR:

   $(\nu\tilde{g})(\nu\tilde{x})(M_1|N) \xrightarrow{(\nu x')\overline{G''}x'} (\nu\tilde{g})(\nu\tilde{x_1})(M_1'|N)$ given $M_1 \xrightarrow{\overline{G}x'} M_1'$, where $\tilde{x_1} = \tilde{x} \setminus \{x'\}$ and $G'' = G \setminus \tilde{g}$. The derivation is given below:

$$\text{PAR: } \frac{M_1 \xrightarrow{\overline{G}x'} M_1'}{M_1|N \xrightarrow{\overline{G}x'} M_1'|N}$$

$$\text{OPEN: } \frac{}{(\nu\tilde{x})(M_1|N) \xrightarrow{(\nu x')\overline{G}x'} (\nu\tilde{x_1})(M_1'|N)}$$

$$\text{GNAME-RES1: } \frac{}{(\nu\tilde{g})(\nu\tilde{x})(M_1|N) \xrightarrow{(\nu x')\overline{G''}x'} (\nu\tilde{g})(\nu\tilde{x_1})(M_1'|N)} \; G'' \neq \emptyset$$

   Since $M_1 \sim M_2$, $M_1 \xrightarrow{\overline{G}x'} M_1'$ means that there is an M_2' such that $M_2 \xrightarrow{\overline{G}x'} M_2'$ and $M_1' \sim M_2'$. Moreover, there exist expressions M'_{N_1} and M'_{N_2} in normal form such that $M_1' \equiv M'_{N_1}$ and $M_2' \equiv M'_{N_2}$. Now, since $M_1' \sim M_2'$, we know $M'_{N_1} \sim M'_{N_2}$. Hence, by construction of S, we can conclude that the pair $((\nu\tilde{g})(\nu\tilde{x_1})(M'_{N_1}|N), (\nu\tilde{g})(\nu\tilde{x_1})(M'_{N_2}|N)) \in S$, and hence $((\nu\tilde{g})(\nu\tilde{x_1})(M_1'|N), (\nu\tilde{g})(\nu\tilde{x_1})(M_2'|N)) \in \equiv S \equiv$.

Comment (1n) Done. Derivation in presence of MOBILITY. Also related to comment (1f) about MOBILITY rule.

8. Case GNAME-RES1, PNAME-RES, MOBILITY:
   $(\nu\tilde{g})(\nu\tilde{x})(M_1|N) \xrightarrow{\mu} (\nu\tilde{g})(\nu\tilde{x})(M_1'|N')$. The derivation is given below:

$$\text{MOBILITY: } \frac{}{M_1|N \xrightarrow{\mu} M_1'|N'}$$

$$\text{PNAME-RES: } \frac{}{(\nu\tilde{x})(M_1|N) \xrightarrow{\mu} (\nu\tilde{x})(M_1'|N')}$$

$$\text{GNAME-RES1: } \frac{}{(\nu\tilde{g})(\nu\tilde{x})(M_1|N) \xrightarrow{\mu} (\nu\tilde{g})(\nu\tilde{x})(M_1'|N')}$$

   and $I(M_1|N) \implies I(M_1'|N')$ for a connectivity invariant I.
   Now consider the following cases for M_1' and N':

(i) $M_1' = M_1$ and N' differs from N only in one of its basic node's interface, i.e. N' is obtained by replacing one basic node $P : G$ in N by $P : G'$, where $G' \subseteq fgn(M_1) \cup fgn(N)$.
Since $M_1 \sim M_2$, $fgn(M_1) = fgn(M_2)$ and $M_1|N \xrightarrow{\mu} M_1|N'$ imply that $M_2|N \xrightarrow{\mu} M_2|N'$ such that $I(M_2|N) \implies I(M_2|N')$, and it can be derived that $(\nu\tilde{g})(\nu\tilde{x})(M_2|N) \xrightarrow{\mu} (\nu\tilde{g})(\nu\tilde{x})(M_2|N')$. Moreover, there exist N'_N in normal form such that $N' \equiv N'_N$. Hence, by construction of S, we can conclude that pair $((\nu\tilde{g})(\nu\tilde{x})(M_1|N'_N), (\nu\tilde{g})(\nu\tilde{x})(M_2|N'_N)) \in S$, and hence $((\nu\tilde{g})(\nu\tilde{x})(M_1|N'), (\nu\tilde{g})(\nu\tilde{x})(M_2|N')) \in \equiv S \equiv$.

(ii) $N' = N$ and M_1' is obtained from M_1 by replacing one of its basic node $P : G_P$ in M_1 by $P : G'_P$ in M_1', where $G'_P \subseteq fgn(M_1) \cup fgn(N)$.
Let M_2 contain a basic node $Q : G_Q$ and M_2' differ from M_2 only due to $Q : G_Q$ replaced by $Q : G'_Q$, where $G'_Q \subseteq fgn(M_2) \cup fgn(N)$.
Consider the following two cases:
(a) G'_P and G'_Q contain gnames only in $fgn(M_1)$ and $fgn(M_2)$, respectively, then M_1' and M_2' can be derived using MOBILITY rule from M_1 and M_2, respectively. Since $M_1 \sim M_2$, $fgn(M_1) = fgn(M_2)$ and $M_1 \xrightarrow{\mu} M_1'$ implies that $M_2 \xrightarrow{\mu} M_2'$, and $M_1' \sim M_2'$.
(b) G'_P and G'_Q also contain gnames in $fgn(N)$. Since the possible new free gnames (other than $fgn(M_1)$ and $fgn(M_2)$), added to basic nodes $P : G_P$ in M_1 and $Q : G_Q$ in M_2 leading to M_1' and M_2', respectively, are drawn from the same set of gnames $fgn(N)$, similarity in behavior (transitions) of M_1' and M_2' is preserved i.e. $M_1' \sim M_2'$.
$M_2|N \xrightarrow{\mu} M_2'|N$ and it can be derived that $(\nu\tilde{g})(\nu\tilde{x})(M_2|N) \xrightarrow{\mu} (\nu\tilde{g})(\nu\tilde{x})(M_2'|N)$ such that $I(M_2|N) \implies I(M_2'|N)$. Moreover, there exist expressions M'_{N_1} and M'_{N_2} in normal form such that $M_1' \equiv M'_{N_1}$ and $M_2' \equiv M'_{N_2}$. Now, since $M_1' \sim M_2'$, we know $M'_{N_1} \sim M'_{N_2}$. Hence, by construction of S, we can conclude that the pair $((\nu\tilde{g})(\nu\tilde{x})(M'_{N_1}|N), (\nu\tilde{g})(\nu\tilde{x})(M'_{N_2}|N)) \in S$, and hence $((\nu\tilde{g})(\nu\tilde{x})(M_1'|N), (\nu\tilde{g})(\nu\tilde{x})(M_2'|N)) \in \equiv S \equiv$.

9. Case GNAME-RES1, PNAME-RES, PAR:

$(\nu\tilde{g})(\nu\tilde{x})(M_1|N) \xrightarrow{\alpha\setminus\tilde{g}} (\nu\tilde{g})(\nu\tilde{x})(M_1'|N)$ given $M_1 \xrightarrow{\alpha} M_1'$. The derivation is given below:

$$\text{PAR: } \frac{M_1 \xrightarrow{\alpha} M_1'}{M_1|N \xrightarrow{\alpha} M_1'|N}\ bn(\alpha) \cap fn(N) = \emptyset$$

$$\text{PNAME-RES: } \frac{M_1|N \xrightarrow{\alpha} M_1'|N}{(\nu\tilde{x})(M_1|N) \xrightarrow{\alpha} (\nu\tilde{x})(M_1'|N)}\ \tilde{x} \cap n(\alpha) = \emptyset$$

$$\text{GNAME-RES1: } \frac{(\nu\tilde{x})(M_1|N) \xrightarrow{\alpha} (\nu\tilde{x})(M_1'|N)}{(\nu\tilde{g})(\nu\tilde{x})(M_1|N) \xrightarrow{\alpha\setminus\tilde{g}} (\nu\tilde{g})(\nu\tilde{x})(M_1'|N)}$$

Since $M_1 \sim M_2$, $M_1 \xrightarrow{\alpha} M_1'$ means that there is an M_2' such that

$M_2 \xrightarrow{\alpha} M_2'$ and $M_1' \sim M_2'$. Moreover, there exist expressions M'_{N_1} and M'_{N_2} in normal form such that $M_1' \equiv M'_{N_1}$ and $M_2' \equiv M'_{N_2}$. Now, since $M_1' \sim M_2'$, we know $M'_{N_1} \sim M'_{N_2}$. Hence, by construction of S, we can conclude that the pair $((\nu\tilde{g})(\nu\tilde{x})(M'_{N_1}|N), (\nu\tilde{g})(\nu\tilde{x})(M'_{N_2}|N)) \in S$, and hence $((\nu\tilde{g})(\nu\tilde{x})(M_1'|N), (\nu\tilde{g})(\nu\tilde{x})(M_2'|N)) \in \equiv S \equiv$.
For the case $\alpha = \mu$, the conditions $I(M_1|N) \implies I(M_1'|N)$ and $I(M_2|N) \implies I(M_2'|N)$, for a connectivity invariant I, also come into effect.

Note that if $\alpha \setminus \tilde{g}$ is of the form $G(x')$ or $z{:}G(x')$, where $x' \in \mathbf{Pn}$, the proof involves following reasoning:
$M_1 \sim M_2$ implies for all $y \in \mathbf{Pn}$, $M_1'\{y/x'\} \sim M_2'\{y/x'\}$. Moreover, there exist expressions M'_{N_1} and M'_{N_2} in normal form such that $M_1' \equiv M'_{N_1}$ and $M_2' \equiv M'_{N_2}$. We infer that for all $y \in \mathbf{Pn}$, $M_1'\{y/x'\} \sim M_2'\{y/x'\}$ implies $M'_{N_1}\{y/x'\} \sim M'_{N_2}\{y/x'\}$. Therefore, for all $y \in \mathbf{Pn}$, $((\nu\tilde{g})(\nu\tilde{x})(M'_{N_1}\{y/x'\}|N), (\nu\tilde{g})(\nu\tilde{x})(M'_{N_2}\{y/x'\}|N)) \in S$. Since $bn(\alpha) \cap fn(N) = \emptyset$, we know $x' \notin fn(N)$. Hence, for all pname $y \in \mathbf{Pn}$, $(\nu\tilde{g})(\nu\tilde{x})(M'_{N_1}\{y/x'\}|N) = ((\nu\tilde{g})(\nu\tilde{x})(M'_{N_1}|N))\{y/x'\}$ and $(\nu\tilde{g})(\nu\tilde{x})(M'_{N_2}\{y/x'\}|N) = ((\nu\tilde{g})(\nu\tilde{x})(M'_{N_2}|N))\{y/x'\}$. Hence, by construction of S, we can conclude that for all $y \in \mathbf{Pn}$, the pair $(((\nu\tilde{g})(\nu\tilde{x})(M'_{N_1}|N))\{y/x'\}, ((\nu\tilde{g})(\nu\tilde{x})(M'_{N_2}|N))\{y/x'\}) \in S$, and hence for all $y \in \mathbf{Pn}$, $(((\nu\tilde{g})(\nu\tilde{x})(M_1'|N)\{y/x'\}), ((\nu\tilde{g})(\nu\tilde{x})(M_2'|N))\{y/x'\}) \in \equiv S \equiv$.

10. Case GNAME-RES1, PNAME-RES, UNI-COM:
$(\nu\tilde{g})(\nu\tilde{x})(M_1|N) \xrightarrow{\tau} (\nu\tilde{g})(\nu\tilde{x})(M_1'|N'\{x'/y\})$ given $M_1 \xrightarrow{\overline{z:G}x'} M_1'$ and $N \xrightarrow{z:G'(y)} N'$. The derivation is given below:

$$\text{UNI-COM:}\ \frac{M_1 \xrightarrow{\overline{z:G}x'} M_1' \quad N \xrightarrow{z:G'(y)} N'}{M_1|N \xrightarrow{\tau} M_1'|N'\{x'/y\}}\ G \cap G' \neq \emptyset$$

$$\text{PNAME-RES:}\ \frac{M_1|N \xrightarrow{\tau} M_1'|N'\{x'/y\}}{(\nu\tilde{x})(M_1|N) \xrightarrow{\tau} (\nu\tilde{x})(M_1'|N'\{x'/y\})}$$

$$\text{GNAME-RES1:}\ \frac{(\nu\tilde{x})(M_1|N) \xrightarrow{\tau} (\nu\tilde{x})(M_1'|N'\{x'/y\})}{(\nu\tilde{g})(\nu\tilde{x})(M_1|N) \xrightarrow{\tau} (\nu\tilde{g})(\nu\tilde{x})(M_1'|N'\{x'/y\})}$$

Since $M_1 \sim M_2$, $M_1 \xrightarrow{\overline{z:G}x'} M_1'$ means that there is an M_2' s.t. $M_2 \xrightarrow{\overline{z:G}x'} M_2'$ and $M_1' \sim M_2'$. Moreover, there exist expressions M'_{N_1}, M'_{N_2} and N'_N in normal form such that $M_1' \equiv M'_{N_1}$, $M_2' \equiv M'_{N_2}$ and $N'\{x'/y\} \equiv N'_N$. Now, since $M_1' \sim M_2'$, we know $M'_{N_1} \sim M'_{N_2}$. Hence, by construction of S, we can conclude that the pair $((\nu\tilde{g})(\nu\tilde{x})(M'_{N_1}|N'_N), (\nu\tilde{g})(\nu\tilde{x})(M'_{N_2}|N'_N)) \in S$, and hence $((\nu\tilde{g})(\nu\tilde{x})(M_1'|N'\{x'/y\}), (\nu\tilde{g})(\nu\tilde{x})(M_2'|N'\{x'/y\})) \in \equiv S \equiv$.

11. Case GNAME-RES1, PNAME-RES, UNI-CLOSE:
$(\nu\tilde{g})(\nu\tilde{x})(M_1|N) \xrightarrow{\tau} (\nu\tilde{g})(\nu\tilde{x}_1)(M_1'|N')$ given $M_1 \xrightarrow{(\nu x')\overline{z:G}x'} M_1'$ and $N \xrightarrow{z:G'(x')} N'$, where $\tilde{x_1} = \tilde{x} \cup \{x'\}$. The derivation is given below:

$$\begin{array}{ll}\text{UNI-CLOSE:} & \dfrac{M_1 \xrightarrow{(\nu x')\overline{z:G}x'} M_1' \quad N \xrightarrow{z:G'(x')} N'}{M_1|N \xrightarrow{\tau} (\nu x')(M_1'|N')} \quad G \cap G' \neq \emptyset \\ \text{PNAME-RES:} & \dfrac{M_1|N \xrightarrow{\tau} (\nu x')(M_1'|N')}{(\nu\tilde{x})(M_1|N) \xrightarrow{\tau} (\nu\tilde{x})(\nu x')(M_1'|N')} \\ \text{GNAME-RES1:} & \dfrac{(\nu\tilde{x})(M_1|N) \xrightarrow{\tau} (\nu\tilde{x})(\nu x')(M_1'|N')}{(\nu\tilde{g})(\nu\tilde{x})(M_1|N) \xrightarrow{\tau} (\nu\tilde{g})(\nu\tilde{x})(\nu x')(M_1'|N')}\end{array}$$

Since $M_1 \sim M_2$, $M_1 \xrightarrow{(\nu x')\overline{z:G}x'} M_1'$ means that there exists an M_2' such that $M_2 \xrightarrow{(\nu x')\overline{z:G}x'} M_2'$ and $M_1' \sim M_2'$. Moreover, there exist expressions M'_{N_1}, M'_{N_2} and N'_N in normal form such that $M_1' \equiv M'_{N_1}$, $M_2' \equiv M'_{N_2}$ and $N' \equiv N'_N$. Now, since $M_1' \sim M_2'$, we know $M'_{N_1} \sim M'_{N_2}$. Hence, by construction of S, we can conclude that the pair $((\nu\tilde{g})(\nu\tilde{x_1})(M'_{N_1}|N'_N), (\nu\tilde{g})(\nu\tilde{x_1})(M'_{N_2}|N'_N)) \in S$, and hence $((\nu\tilde{g})(\nu\tilde{x_1})(M_1'|N'), (\nu\tilde{g})(\nu\tilde{x_1})(M_2'|N')) \in \equiv S \equiv$.

12. Case GNAME-RES1, PNAME-RES, COM:
$(\nu\tilde{g})(\nu\tilde{x})(M_1|N) \xrightarrow{\overline{G''}x'} (\nu\tilde{g})(\nu\tilde{x})(M_1'|N'\{x'/y\})$ given $M_1 \xrightarrow{\overline{G}x'} M_1'$ and $N \xrightarrow{G'(y)} N'$, where $G'' = G \setminus \tilde{g}$. The derivation is given below:

$$\begin{array}{lll}\text{COM:} & \dfrac{M_1 \xrightarrow{\overline{G}x'} M_1' \quad N \xrightarrow{G'(y)} N'}{M_1|N \xrightarrow{\overline{G}x'} M_1'|N'\{x'/y\}} & G \cap G' \neq \emptyset \\ \text{PNAME-RES:} & \dfrac{M_1|N \xrightarrow{\overline{G}x'} M_1'|N'\{x'/y\}}{(\nu\tilde{x})(M_1|N) \xrightarrow{\overline{G}x'} (\nu\tilde{x})(M_1'|N'\{x'/y\})} & x' \notin \tilde{x} \\ \text{GNAME-RES1:} & \dfrac{(\nu\tilde{x})(M_1|N) \xrightarrow{\overline{G}x'} (\nu\tilde{x})(M_1'|N'\{x'/y\})}{(\nu\tilde{g})(\nu\tilde{x})(M_1|N) \xrightarrow{\overline{G''}x'} (\nu\tilde{g})(\nu\tilde{x})(M_1'|N'\{x'/y\})} & G'' \neq \emptyset\end{array}$$

Since $M_1 \sim M_2$, $M_1 \xrightarrow{\overline{G}x'} M_1'$ means that there exists an M_2' such that $M_2 \xrightarrow{\overline{G}x'} M_2'$ and $M_1' \sim M_2'$. Moreover, there exist expressions M'_{N_1}, M'_{N_2} and N'_N in normal form such that $M_1' \equiv M'_{N_1}$, $M_2' \equiv M'_{N_2}$ and $N'\{x'/y\} \equiv N'_N$. Now, since $M_1' \sim M_2'$, we know $M'_{N_1} \sim M'_{N_2}$. Hence, by construction of S, we can conclude that the pair $((\nu\tilde{g})(\nu\tilde{x})(M'_{N_1}|N'_N), (\nu\tilde{g})(\nu\tilde{x})(M'_{N_2}|N'_N)) \in S$, and hence $((\nu\tilde{g})(\nu\tilde{x})(M_1'|N'\{x'/y\}), (\nu\tilde{g})(\nu\tilde{x})(M_2'|N'\{x'/y\})) \in \equiv S \equiv$.

13. Case GNAME-RES1, PNAME-RES, COM-RES:
$(\nu\tilde{g})(\nu\tilde{x})(M_1|N) \xrightarrow{(\nu x')\overline{G''}x'} (\nu\tilde{g})(\nu\tilde{x})(M_1'|N')$ given $M_1 \xrightarrow{(\nu x')\overline{G}x'} M_1'$ and $N \xrightarrow{G'(x')} N'$, where $G'' = G \setminus \tilde{g}$. The derivation is given below:

$$\text{COM-RES:}\ \frac{M_1 \xrightarrow{(\nu x')\overline{G}x'} M_1' \quad N \xrightarrow{G'(x')} N'}{M_1|N \xrightarrow{(\nu x')\overline{G}x'} M_1'|N'} \quad G \cap G' \neq \emptyset$$

$$\text{PNAME-RES:}\ \frac{M_1|N \xrightarrow{(\nu x')\overline{G}x'} M_1'|N'}{(\nu\tilde{x})(M_1|N) \xrightarrow{(\nu x')\overline{G}x'} (\nu\tilde{x})(M_1'|N')} \quad x' \notin \tilde{x}$$

$$\text{GNAME-RES1:}\ \frac{(\nu\tilde{x})(M_1|N) \xrightarrow{(\nu x')\overline{G}x'} (\nu\tilde{x})(M_1'|N')}{(\nu\tilde{g})(\nu\tilde{x})(M_1|N) \xrightarrow{(\nu x')\overline{G''}x'} (\nu\tilde{g})(\nu\tilde{x})(M_1'|N')} \quad G'' \neq \emptyset$$

Since $M_1 \sim M_2$, $M_1 \xrightarrow{(\nu x')\overline{G}x'} M_1'$ means that there is an M_2' such that $M_2 \xrightarrow{(\nu x')\overline{G}x'} M_2'$ and $M_1' \sim M_2'$. Moreover, there exist expressions M_{N_1}', M_{N_2}' and N_N' in normal form such that $M_1' \equiv M_{N_1}'$, $M_2' \equiv M_{N_2}'$ and $N' \equiv N_N'$. Now, since $M_1' \sim M_2'$, we know $M_{N_1}' \sim M_{N_2}'$. Hence, by construction of S, we can conclude that the pair $((\nu\tilde{g})(\nu\tilde{x})(M_{N_1}'|N_N'), (\nu\tilde{g})(\nu\tilde{x})(M_{N_2}'|N_N')) \in S$, and hence $((\nu\tilde{g})(\nu\tilde{x})(M_1'|N'), (\nu\tilde{g})(\nu\tilde{x})(M_2'|N')) \in \equiv S \equiv$.

14. Case GNAME-RES2, PNAME-RES, COM:

$(\nu\tilde{g})(\nu\tilde{x})(M_1|N) \xrightarrow{\tau} (\nu\tilde{g})(\nu\tilde{x})(M_1'|N'\{x'/y\})$ given $M_1 \xrightarrow{\overline{G}x'} M_1'$ and $N \xrightarrow{G'(y)} N'$. The derivation is given below:

$$\text{COM:}\ \frac{M_1 \xrightarrow{\overline{G}x'} M_1' \quad N \xrightarrow{G'(y)} N'}{M_1|N \xrightarrow{\overline{G}x'} M_1'|N'\{x'/y\}} \quad G \cap G' \neq \emptyset$$

$$\text{PNAME-RES:}\ \frac{M_1|N \xrightarrow{\overline{G}x'} M_1'|N'\{x'/y\}}{(\nu\tilde{x})(M_1|N) \xrightarrow{\overline{G}x'} (\nu\tilde{x})(M_1'|N'\{x'/y\})} \quad x' \notin \tilde{x}$$

$$\text{GNAME-RES2:}\ \frac{(\nu\tilde{x})(M_1|N) \xrightarrow{\overline{G}x'} (\nu\tilde{x})(M_1'|N'\{x'/y\})}{(\nu\tilde{g})(\nu\tilde{x})(M_1|N) \xrightarrow{\tau} (\nu\tilde{g})(\nu\tilde{x})(M_1'|N'\{x'/y\})} \quad G \setminus \tilde{g} = \emptyset$$

Since $M_1 \sim M_2$, $M_1 \xrightarrow{\overline{G}x'} M_1'$ means that there is an M_2' such that $M_2 \xrightarrow{\overline{G}x'} M_2'$ and $M_1' \sim M_2'$. Moreover, there exist expressions M_{N_1}', M_{N_2}' and N_N' in normal form such that $M_1' \equiv M_{N_1}'$, $M_2' \equiv M_{N_2}'$ and $N'\{x'/y\} \equiv N_N'$. Now, since $M_1' \sim M_2'$, we know $M_{N_1}' \sim M_{N_2}'$. Hence, by construction of S, we can conclude that the pair $((\nu\tilde{g})(\nu\tilde{x})(M_{N_1}'|N_N'), (\nu\tilde{g})(\nu\tilde{x})(M_{N_2}'|N_N')) \in S$, and hence $((\nu\tilde{g})(\nu\tilde{x})(M_1'|N'\{x'/y\}), (\nu\tilde{g})(\nu\tilde{x})(M_2'|N'\{x'/y\})) \in \equiv S \equiv$.

15. Case GNAME-RES2, PNAME-RES, PAR:

$(\nu\tilde{g})(\nu\tilde{x})(M_1|N) \xrightarrow{\tau} (\nu\tilde{g})(\nu\tilde{x})(M_1'|N)$ given $M_1 \xrightarrow{\overline{G}x'} M_1'$.

$$\text{PAR:}\ \frac{M_1 \xrightarrow{\overline{G}x'} M_1'}{M_1|N \xrightarrow{\overline{G}x'} M_1'|N}$$

$$\text{PNAME-RES:}\ \frac{M_1|N \xrightarrow{\overline{G}x'} M_1'|N}{(\nu\tilde{x})(M_1|N) \xrightarrow{\overline{G}x'} (\nu\tilde{x})(M_1'|N)} \quad x' \notin \tilde{x}$$

$$\text{GNAME-RES2:}\ \frac{(\nu\tilde{x})(M_1|N) \xrightarrow{\overline{G}x'} (\nu\tilde{x})(M_1'|N)}{(\nu\tilde{g})(\nu\tilde{x})(M_1|N) \xrightarrow{\tau} (\nu\tilde{g})(\nu\tilde{x})(M_1'|N)} \quad G \setminus \tilde{g} = \emptyset$$

Since $M_1 \sim M_2$, $M_1 \xrightarrow{\overline{G}x'} M'_1$ means that there exists an M'_2 such that $M_2 \xrightarrow{\overline{G}x'} M'_2$ and $M'_1 \sim M'_2$. Moreover, there exist expression M'_{N_1} and M'_{N_2} in normal form such that $M'_1 \equiv M'_{N_1}$ and $M'_2 \equiv M'_{N_2}$. Now, since $M'_1 \sim M'_2$, we know $M'_{N_1} \sim M'_{N_2}$. Hence, by construction of S, we can conclude that the pair $((\nu\tilde{g})(\nu\tilde{x})(M'_{N_1}|N), (\nu\tilde{g})(\nu\tilde{x})(M'_{N_2}|N)) \in S$, and hence $((\nu\tilde{g})(\nu\tilde{x})(M'_1|N), (\nu\tilde{g})(\nu\tilde{x})(M'_2|N)) \in \equiv S \equiv$.

By considering the 15 cases and their symmetric counterparts due to commutativity of '|' operator, all possible derivations are covered and we conclude that for every transition from $(\nu\tilde{g})(\nu\tilde{x})(M_1|N)$, there is a transition from $(\nu\tilde{g})(\nu\tilde{x})(M_2|N)$ such that the destinations of the two transitions are related by $\equiv S \equiv$. Thus we establish that S is a strong bisimulation upto $\equiv$. Following Lemma 4, we conclude that S is a strong bisimulation. Therefore, $\sim$ is preserved by restriction of pnames and gnames, and the parallel operator for ω-expressions in normal form.

This proof is *complete* because at each proof step all possible transitions from an expression are considered to find its derivatives. The fifteen cases along with their symmetric counterparts for the parallel operator cover all the derivation possibilities. All the possible transitions at the node level (pertaining to broadcast, unicast, silent action, and mobility) are taken into account through the derivations given in the proof. □

B. Symbolic Bisimulation for the ω_0-Calculus

We prove that the symbolic bisimulation equivalence for the ω_0-calculus is a congruence. The proof for the extended calculi follow along the same lines.

Lemma 15. *For all $M_1, M_2 \in \mathbf{N_{nf}}$, i.e., M_1, M_2 are in normal form, the following hold:*

(i) $M_1 \asymp M_2$ implies $\forall g \in \mathbf{Gn} : (\nu g)M_1 \asymp (\nu g)M_2$; and

(ii) $M_1 \asymp M_2$ implies $\forall N \in \mathbf{N_{nf}}$: $M_1|N \asymp M_2|N$.

Proof. We show parts *(i–ii)* of the lemma simultaneously by considering the set $S = \{((\nu\tilde{g})(M_1|N), (\nu\tilde{g})(M_2|N)) \mid M_1 \asymp M_2, \tilde{g} \subseteq \mathbf{Gn}, M_1, M_2, N \in \mathbf{N_{nf}}\}$. Following Lemma 4 it is sufficient to show that S is a strong bisimulation upto $\equiv$ to establish this lemma.

Note that if $M_1 \asymp M_2$ then $fgn(M_1) = fgn(M_2)$, and hence $fgn((\nu\tilde{g})(M_1|N)) = fgn((\nu\tilde{g})(M_2|N))$ for all $\tilde{g}$ and N. We then show that every transition from $(\nu\tilde{g})(M_1|N)$ can be matched by $(\nu\tilde{g})(M_2|N)$ by considering the derivations of transitions. Transitions for $(\nu\tilde{g})(M_1|N)$ can be derived by the use of rules GNAME-RES1, GNAME-RES2, MOBILITY, PAR and COM. Only the last two steps of each transition derivation are considered in the proof. Most importantly, following Lemma 7, we do not need to consider derivations that use STRUCT rules in the last step. From the structural operational semantics, the last step of a derivation will be due to the outermost $(\nu\tilde{g})$ in the expression, and the first step due to the parallel composition $(M_1|N)$. We omit in the proof the symmetric cases arising due to the commutativity of the parallel operator

'|'. This gives rise to 5 cases (combinations of rules in the last two steps in a derivation).

1. Case GNAME-RES1, COM:

   $(\nu\tilde{g})(M_1|N) \overset{C_1\wedge C,\overline{G''}x}{\longrightarrow} (\nu\tilde{g})(M_1'|N'\{x/y\})$ given $M_1 \overset{C_1,\overline{G}x}{\longrightarrow} M_1'$ and $N \overset{C,G'(y)}{\longrightarrow} N'$, where $G'' = G \setminus \tilde{g}$. The derivation is given below:

   $$\text{COM:}\ \frac{M_1 \overset{C_1,\overline{G}x}{\longrightarrow} M_1' \quad N \overset{C,G'(y)}{\longrightarrow} N'}{M_1|N \overset{C_1\wedge C,\overline{G}x}{\longrightarrow} M_1'|N'\{x/y\}}\ G\cap G' \neq \emptyset$$
   $$\text{GNAME-RES1:}\ \frac{M_1|N \overset{C_1\wedge C,\overline{G}x}{\longrightarrow} M_1'|N'\{x/y\}}{(\nu\tilde{g})(M_1|N) \overset{C_1\wedge C,\overline{G''}x}{\longrightarrow} (\nu\tilde{g})(M_1'|N'\{x/y\})}\ G''\neq\emptyset$$

   Since $M_1 \asymp M_2$, $M_1 \overset{C_1,\overline{G}x}{\longrightarrow} M_1'$ implies $\exists M_2'$, β, and C_2 such that $M_2 \overset{C_2,\beta}{\longrightarrow} M_2'$ and $C_1 \triangleright C_2$, $\overline{G}x\sigma_{C_1} \equiv \beta\sigma_{C_1}$, $M_1'\sigma_{C_1} \asymp M_2'\sigma_{C_1}$. Moreover, there exist expressions M'_{N_1}, M'_{N_2} and N'_N in normal form such that $M_1' \equiv M'_{N_1}$, $M_2' \equiv M'_{N_2}$ and $N'\{x/y\} \equiv N'_N$. Now, since $M_1'\sigma_{C_1} \asymp M_2'\sigma_{C_1}$, we know $M'_{N_1}\sigma_{C_1} \asymp M_{N_2}\sigma'_{C_1}$. Hence, by construction of S, we can conclude that the pair $((\nu\tilde{g})(M'_{N_1}\sigma_{C_1}|N'_N\sigma_{C_1\wedge C}), (\nu\tilde{g})(M'_{N_2}\sigma_{C_1}|N'_N\sigma_{C_1\wedge C})) \in S$, and hence $((\nu\tilde{g})(M_1'|N'\{x/y\})\sigma_{C_1\wedge C}, (\nu\tilde{g})(M_2'|N'\{x/y\})\sigma_{C_1\wedge C}) \in \equiv S \equiv$.

2. Case GNAME-RES1, MOBILITY:

   Similar to Comment (1n) for proof of Lemma 8.

   $(\nu\tilde{g})(M_1|N) \overset{\mathbf{true},\mu}{\longrightarrow} (\nu\tilde{g})(M_1'|N')$. The derivation is given below:

   $$\text{MOBILITY:}\ \frac{}{M_1|N \overset{\mathbf{true},\mu}{\longrightarrow} M_1'|N'}$$
   $$\text{GNAME-RES1:}\ \frac{M_1|N \overset{\mathbf{true},\mu}{\longrightarrow} M_1'|N'}{(\nu\tilde{g})(M_1|N) \overset{\mathbf{true},\mu}{\longrightarrow} (\nu\tilde{g})(M_1'|N')}$$

   and $I(M_1|N) \implies I(M_1'|N')$ for a connectivity invariant I.
   A case analysis of M_1' and N', similar to as in Case 8 (GNAME-RES1, PNAME-RES, MOBILITY) for proof of Lemma 8 given in Appendix A, can be used to conclude that $((\nu\tilde{g})(M_1'|N'), (\nu\tilde{g})(M_2'|N')) \in \equiv S \equiv$.

3. Case GNAME-RES1, PAR:

   $(\nu\tilde{g})(M_1|N) \overset{C_1,\alpha\setminus\tilde{g}}{\longrightarrow} (\nu\tilde{g})(M_1'|N)$ given $M_1 \overset{C_1,\alpha}{\longrightarrow} M_1'$. The derivation is given below:

   $$\text{PAR:}\ \frac{M_1 \overset{C_1,\alpha}{\longrightarrow} M_1'}{M_1|N \overset{C_1,\alpha}{\longrightarrow} M_1'|N}\ bn(\alpha)\cap fn(N) = \emptyset$$
   $$\text{GNAME-RES1:}\ \frac{M_1|N \overset{C_1,\alpha}{\longrightarrow} M_1'|N}{(\nu\tilde{g})(M_1|N) \overset{C_1,\alpha\setminus\tilde{g}}{\longrightarrow} (\nu\tilde{g})(M_1'|N)}$$

   Since $M_1 \asymp M_2$, $M_1 \overset{C_1,\alpha}{\longrightarrow} M_1'$ implies $\exists M_2'$, β, and C_2 such that $M_2 \overset{C_2,\beta}{\longrightarrow} M_2'$ and $C_1 \triangleright C_2$, $\alpha\sigma_{C_1} \equiv \beta\sigma_{C_1}$, $M_1'\sigma_{C_1} \asymp M_2'\sigma_{C_1}$. More-

over, there exist expressions M'_{N_1} and M'_{N_2} in normal form such that $M'_1 \equiv M'_{N_1}$ and $M'_2 \equiv M'_{N_2}$. Now, since $M'_1\sigma_{C_1} \asymp M'_2\sigma_{C_1}$, we know $M'_{N_1}\sigma_{C_1} \asymp M'_{N_2}\sigma_{C_1}$. Hence, by construction of S, we can conclude that the pair $((\nu\tilde{g})(M'_{N_1}\sigma_{C_1}|N\sigma_{C_1}), (\nu\tilde{g})(M'_{N_2}\sigma_{C_1}|N\sigma_{C_1})) \in S$, and hence $((\nu\tilde{g})(M'_1|N)\sigma_{C_1}, (\nu\tilde{g})(M'_2|N)\sigma_{C_1}) \in \equiv S \equiv$.
For the case $\alpha = \mu$, the conditions $I(M_1|N) \implies I(M'_1|N)$ and $I(M_2|N) \implies I(M'_2|N)$, for a connectivity invariant I, also come into effect in the above derivations.

For the case when $\alpha \setminus \tilde{g}$ is of the form $G(x')$, we can reason in a manner similar to that for the Case 9 (GNAME-RES1, PNAME-RES, PAR) for proof of Lemma 8 given in Appendix A.

4. Case GNAME-RES2, COM:

   $(\nu\tilde{g})(M_1|N) \xrightarrow{C_1\wedge C,\tau} (\nu\tilde{g})(M'_1|N'\{x/y\})$ given $M_1 \xrightarrow{C_1,\overline{G}x} M'_1$ and $N \xrightarrow{C,G'(y)} N'$. The derivation is given below:

$$\text{COM:}\ \frac{M_1 \xrightarrow{C_1,\overline{G}x} M'_1 \quad N \xrightarrow{C,G'(y)} N'}{M_1|N \xrightarrow{C_1\wedge C,\overline{G}x} M'_1|N'\{x/y\}}\ G\cap G' \neq \emptyset$$

$$\text{GNAME-RES2:}\ \frac{M_1|N \xrightarrow{C_1\wedge C,\overline{G}x} M'_1|N'\{x/y\}}{(\nu\tilde{g})(M_1|N) \xrightarrow{C_1\wedge C,\tau} (\nu\tilde{g})(M'_1|N'\{x/y\})}\ G\setminus\tilde{g} = \emptyset$$

   Since $M_1 \asymp M_2$, $M_1 \xrightarrow{C_1,\overline{G}x} M'_1$ implies $\exists M'_2$, β, and C_2 such that $M_2 \xrightarrow{C_2,\beta} M'_2$ and $C_1 \triangleright C_2$, $\overline{G}x\sigma_{C_1} \equiv \beta\sigma_{C_1}$, $M'_1\sigma_{C_1} \asymp M'_2\sigma_{C_1}$. Moreover, there exist expressions M'_{N_1}, M'_{N_2} and N'_N in normal form such that $M'_1 \equiv M'_{N_1}$, $M'_2 \equiv M'_{N_2}$ and $N'\{x/y\} \equiv N'_N$. Now, since $M'_1\sigma_{C_1} \asymp M'_2\sigma_{C_1}$, we know $M'_{N_1}\sigma_{C_1} \asymp M'_{N_2}\sigma_{C_1}$. Hence, by construction of S, we can conclude that $((\nu\tilde{g})(M'_{N_1}\sigma_{C_1}|N'_N\sigma_{C_1\wedge C}), (\nu\tilde{g})(M'_{N_2}\sigma_{C_1}|N'_N\sigma_{C_1\wedge C})) \in S$, and hence $((\nu\tilde{g})(M'_1|N'\{x/y\})\sigma_{C_1\wedge C}, (\nu\tilde{g})(M'_2|N'\{x/y\})\sigma_{C_1\wedge C}) \in \equiv S \equiv$.

5. Case GNAME-RES2, PAR:

   $(\nu\tilde{g})(M_1|N) \xrightarrow{C_1,\tau} (\nu\tilde{g})(M'_1|N)$ given $M_1 \xrightarrow{C_1,\overline{G}x} M'_1$. The derivation is given below:

$$\text{PAR:}\ \frac{M_1 \xrightarrow{C_1,\overline{G}x} M'_1}{M_1|N \xrightarrow{C_1,\overline{G}x} M'_1|N}$$

$$\text{GNAME-RES2:}\ \frac{M_1|N \xrightarrow{C_1,\overline{G}x} M'_1|N}{(\nu\tilde{g})(M_1|N) \xrightarrow{C_1,\tau} (\nu\tilde{g})(M'_1|N)}\ G\setminus\tilde{g} = \emptyset$$

   Since $M_1 \asymp M_2$, $M_1 \xrightarrow{C_1,\overline{G}x} M'_1$ implies $\exists M'_2$, β, and C_2 such that $M_2 \xrightarrow{C_2,\beta} M'_2$ and $C_1 \triangleright C_2$, $\overline{G}x\sigma_{C_1} \equiv \beta\sigma_{C_1}$, $M'_1\sigma_{C_1} \asymp M'_2\sigma_{C_1}$. Moreover, there exist expression M'_{N_1} and M'_{N_2} in normal form such that $M'_1 \equiv M'_{N_1}$ and $M'_2 \equiv M'_{N_2}$. Now, since $M'_1\sigma_{C_1} \asymp M'_2\sigma_{C_1}$, we

know $M'_{N_1}\sigma_{C_1} \asymp M'_{N_2}\sigma_{C_1}$. Hence, by construction of S, we can conclude that $((\nu\tilde{g})(M'_{N_1}\sigma_{C_1}|N\sigma_{C_1}), (\nu\tilde{g})(M'_{N_2}\sigma_{C_1}|N\sigma_{C_1})) \in S$, and hence $((\nu\tilde{g})(M'_1|N)\sigma_{C_1}, (\nu\tilde{g})(M'_2|N)\sigma_{C_1}) \in \equiv S \equiv$.

By considering the 5 cases and their symmetric counterparts due to the commutativity of '|' operator, all possible derivations are covered and we conclude that S is a symbolic bisimulation up to $\equiv$. Following Lemma 4 we conclude that S is a symbolic bisimulation. Therefore, $\asymp$ is preserved by restriction of gnames and the parallel operator for ω_0-expressions in normal form.

This proof is *complete* because at each proof step all possible transitions from an expression are considered to find its derivatives. The five cases along with their symmetric counterparts for the parallel operator cover all the derivation possibilities. All the possible transitions at the node level (pertaining to broadcast send/receive, silent action, and mobility) are taken into account through the derivations given in the proof. □

Theorem 16 (Congruence for Symbolic Bisimulation for the ω_0-Calculus). *$\asymp$ is a congruence for the ω_0-calculus; i.e., for all $M_1, M_2 \in \mathbf{N}$, the following hold:*

(i) $M_1 \asymp M_2$ implies $\forall g \in \mathbf{Gn} : (\nu g)M_1 \asymp (\nu g)M_2$; and
(ii) $M_1 \asymp M_2$ implies $\forall N \in \mathbf{N} : M_1|N \asymp M_2|N$.

Proof: Let $M_1 \equiv M_{N_1}$ and $M_2 \equiv M_{N_2}$, where M_{N_1} and M_{N_2} are in normal form. Then the following holds:

- $M_1 \asymp M_2$ implies $M_{N_1} \asymp M_{N_2}$ (from Definition 2 and Lemma 4). $M_{N_1} \asymp M_{N_2}$ implies $\forall g \in \mathbf{G_n}$: $(\nu g)M_{N_1} \asymp (\nu g)M_{N_2}$ (by Lemma 15), which in turn implies $(\nu g)M_1 \asymp (\nu g)M_2$ (by Def. 2 and Lemma 4). Therefore, whenever $M_1 \asymp M_2$ then $(\nu g)M_1 \asymp (\nu g)M_2$.
- $M_1 \asymp M_2$ implies $M_{N_1} \asymp M_{N_2}$ (from Definition 2 and Lemma 4). $M_{N_1} \asymp M_{N_2}$ implies for any $N \in \mathbf{N}$, and $N \equiv N_N$ where, $N_N \in \mathbf{N_{nf}}$: $(M_{N_1}|N_N) \asymp (M_{N_2}|N_N)$ (by Lemma 15), which in turn implies $(M_1|N) \asymp (M_2|N)$ (by Def. 2 and Lemma 4). Therefore, whenever $M_1 \asymp M_2$ then $(M_1|N) \asymp (M_2|N)$.

$\asymp$ is preserved by all the node contexts for the ω_0-calculus. Hence, $\asymp$ is a congruence for the ω_0-calculus. □