Fundamenta Informaticae XX (2010) 1–41
IOS Press

Equational Reasoning on Mobile Ad Hoc Networks

Fatemeh Ghassemi
Sharif University of Technology,
Tehran, Iran
fghassemi@mehr.sharif.edu

Wan Fokkink
Vrije Universiteit,
Amsterdam, The Netherlands
wanf@cs.vu.nl

Ali Movaghar
Sharif University of Technology,
Tehran, Iran
movaghar@sharif.edu

Abstract. We provide an equational theory for *Restricted Broadcast Process Theory* to reason about ad hoc networks. We exploit an extended algebra called *Computed Network Theory* to axiomatize restricted broadcast. It allows one to define the behavior of an ad hoc network with respect to the underlying topologies. We give a sound and ground-complete axiomatization for *CNT* terms with finite-state behavior, modulo what we call rooted branching computed network bisimilarity.

1. Introduction

In Mobile Ad hoc Networks (MANETs), nodes communicate directly with each other using wireless transceivers (possibly along multihop paths) without the need for a fixed infrastructure. The primitive means of communication in MANETs is local broadcast; only nodes located in the range of a transmitter receive data. Thus nodes participate in a broadcast according to the underlying topology of nodes. On the other hand, nodes move arbitrarily, and the topology of the network changes dynamically. Local broadcast and topology changes are the main modeling challenges in MANETs.

We introduced *Restricted Broadcast Process Theory* (*RBPT*) in [6], to specify and verify ad hoc networks, taking into account mobility. *RBPT* specifies a MANET by composing nodes using a restricted

(local) broadcast operator, and by specifying a protocol deployed at a node using a process algebraic notation. Topology changes are modeled implicitly in the semantics, and thus one can verify a network with respect to different topology changes. An advantage of *RBPT* compared to similar algebras is that the specification of a MANET does not include any specification about changes of underlying topologies. The behavior of an ad hoc network is equivalent to its behaviors with respect to all possible topologies. We provided an equational system to reason about *RBPT* terms in [7], which is complete for the recursion-free part of *RBPT*. Our axiomatization borrows from the process algebra *ACP* [3] auxiliary (left merge and communication merge) operators to axiomatize the interleaving behavior of parallel composition. These equations need to take into account not only their observational behaviors, but also the set of topologies for which such behaviors are observed. To this aim, we first extended *RBPT* with new terms, called *Computed Network Theory* (*CNT*), because the extended terms contain a specification of a set of topologies, and their observed behavior is computed with respect to those topologies. Network restrictions on the underlying topology are expressed explicitly in the syntax. The operational semantics of *CNT* is given by constrained labeled transition systems, in which the transitions are subscripted by a set of network restrictions.

In this paper, to illustrate the applicability of our framework in the verification of MANETs, we extend *RBPT* with new operators for verification purposes: *encapsulation* and *abstraction*. The encapsulation operator, parameterized by a set of messages, disallows communications on a set of messages. The *abstraction* or *hide* operator, hides communications on a set of messages from the external observer by turning them into the unobservable action τ. We extend our equational system given in [7] with axioms for the new operators (encapsulation, abstraction and recursion), and two proof principles to reason about recursive behaviors with finite-state models. We introduce a behavioral equivalence relation, so-called *rooted branching computed network bisimilarity*, which we prove to be an equivalence relation and a congruence with respect to all *CNT* operators. Then we provide a sound axiomatization of *CNT* modulo rooted branching computed network bisimilarity, which is ground-complete for *CNT* terms with finite-state behaviors. We prove that a *CNT* term has finite-state behavior if its syntax is *essentially finite-state*. The application of our equational system is illustrated with a simple routing protocol.

The reminder of the paper is structured as follows. First it is explained how we model MANETs in Section 2. In Section 3 we briefly explain *RBPT*, and then introduce *CNT*, an extension of *RBPT*, in Section 4. We present the semantics of *CNT* in Section 5, and develop our behavioral equivalence relation on *CNT* terms in Section 6. In Section 7, we provide a sound axiomatization for *CNT*. In Section 8, we show that our axiomatization is ground-complete for a subset of *CNT* terms with finite-state behaviors. We give an overview on related work in Section 9, and finally we present our conclusions and future plans in Section 10. The proofs are presented in the appendix.

2. Concepts for Modeling MANETs

In this section, it is explained how we model mobility, the dynamics of the underlying topology, and local broadcast communication [6, 7].

In wireless communication, nodes are equipped with wireless transceivers, by which they send and receive information. For each (sender) node, a transmission range is considered, which is an area in which the strength of emitted signals (of data) from the node is strong enough to be sensed by other nodes. The transmission range is not the same for different nodes and it depends on the power used to

emit the signal. A node B is *connected to* a node A, if B is located within the transmission range of A. It is said that B is in the vicinity of A. This connectivity relation between nodes, which is not necessarily symmetric, introduces a *topology* concept. Let Loc denote a finite set of logical addresses, ranged over by ℓ, which models the hardware addresses of nodes. Moreover, A, B, C denote concrete addresses. A topology is a function $\gamma : Loc \rightarrow I\!P(Loc)$, where $\gamma(\ell)$ denotes the set of nodes connected to ℓ. This function models unidirectional connectivity between nodes.

Since nodes in MANETs are mobile, the underlying topology changes. There are two approaches in modeling of topology changes; in one approach, mobility is modeled explicitly in the specification, like in [5, 20, 11, 26, 13], while in the other approach, it is modeled in the semantics [24, 6]. The latter approach provides us with a natural way to model and verify MANETs, since the mobility specification is not a part of MANET protocols. Similarly there are two approaches to model mobility in the semantics; one models it explicitly by defining a set of mobility rules, while the other models node mobility implicitly. In *explicit* modeling of mobility in the semantics, the underlying topology is modeled as a part of the semantics state, and mobility is modeled by performing transitions (with an unobservable action) between states, by the application of mobility rules which manipulate the topology model. In *implicit* modeling, each state is a representative of all possible topologies a network can meet and a network can be at any of these topologies. We have modeled mobility implicitly in the semantics, since this approach releases us from encoding the underlying topology as a part of the specification, which also allows modular specification of MANETs.

When a node broadcasts, all nodes in its vicinity are potential receivers. However, local broadcast is non-blocking and lossy, which means the sender broadcasts irrespective of who is going to receive, and a receiver may lose the message due to signal interference. If two nodes with a common vicinity broadcast simultaneously, the emitted signals may interfere at the receiver. MAC-layer protocols, at each node, are responsible to prevent such interferences in MANETs. However, interferences cannot be avoided completely; for instance IEEE 802.11 exploits a schema to reduce interferences, but does not offer any MAC-layer recovery on broadcast frames. We model unreliable local broadcast in our calculus, but we abstract away from interferences and only consider successful receive actions. Thus our framework is only suitable to specify MANET protocols above the MAC-layer. The process calculi dealing with interferences in wireless systems are [22, 21], while the latter is in a timed setting.

The modeling approach of the underlying topology and mobility affects the definition of the local broadcast semantics. Generally speaking, when mobility is modeled explicitly in the semantics, the behavior of a network in a local broadcast communication is defined in terms of the underlying topology; the (subset of) nodes that are connected to the sender will take part in the communication. However, in our approach, the behavior of the network defines the set of topologies under which such behavior is possible. Consequently our approach results in a compact labeled transition system, as illustrated in Figure 1. Since our broadcast is unreliable, a behavior is possible for the set of topologies in which nodes that have participated in the communication should be connected to the sender, while the other connections between nodes can be anything. We introduce network restrictions to formally specify the set of topologies involved in a transition. Their computation can be automated easily, which is helpful for verifying an application using e.g. model checking. Furthermore we exploit them to axiomatize our algebra.

Let an unknown address be represented by $?$. The unknown address is used in the semantics of a receiving node. Furthermore, an unknown address is used for a node whose address is concealed from an external observer. The set of addresses extended with the unknown address is denoted as $Loc_?$, which by

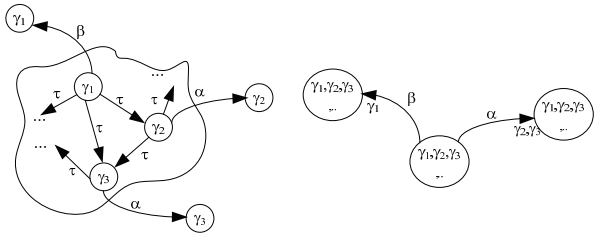

Figure 1. Comparison of the semantics models of local broadcast communication in explicit (at the left) and implicit (at the right) modeling of mobility.

abuse of notation is also ranged over by ℓ. We assume a binary relation $\rightsquigarrow$ on $Loc \times Loc_?$, which imposes connection relations between addresses. The direction of relation explains the direction of data flow in a transmission. A relation $? \rightsquigarrow A$ denotes that a node with logical address A should be in the range of a node with an unknown address, while $A \rightsquigarrow B$ denotes that a node with address B is connected to a node with address A, or A can transmits data to B. The relation $\rightsquigarrow$ need not be symmetric and transitive. By default each node is connected to itself: $\ell \rightsquigarrow \ell$. A *network restriction* is a set of relations $\ell \rightsquigarrow \ell'$. The network restriction $C[B/A]$ is obtained from the network restriction C by substituting B for A in C. Each network restriction C represents the set of topologies that satisfy the relations in C. In particular, the empty network restriction $\{\}$ denotes all possible topologies.

3. Restricted Broadcast Process Theory

Network protocols (in particular MANET protocols) rely on data. To separate the manipulation of data from processes, we make use of equational abstract data types [4]. Data is specified by equational specifications: one can declare data types (so-called *sorts*) and functions working upon these data types, and describe the meaning of these functions by equational axioms. Following the approach of [16, 18], we consider Restricted Broadcast Process Theory with equational abstract data types. The semantics of the data part (of a specification) is defined the same way as in [18]. It should contain the $Bool$ domain with distinct T and F constants.

Before going through the formal syntax definitions of *RBPT*, we define some notations applied in these definitions. Let V denote a countably infinite set of data variables ranged over by x, y, and D a data sort name. Let u and w range over closed and open data terms, respectively. We use $\widehat{w}$ and $\langle x : D\rangle$ to denote finite sequences $w_1, \ldots, w_k$ and $x_1 : D_1, \ldots, x_k : D_k$ for some $k \in I\!N$, $|\widehat{w}|$ and $|\langle x : D\rangle|$ for their length k, and $\{\widehat{w}/\widehat{x}\}$ for simultaneous substitutions $\{w_1/x_1\}, \ldots, \{w_k/x_k\}$. Let Msg denote a set of message types communicated over a network and ranged over by m. For each message type m, $domain_m : I\!P(D_1 \times \ldots \times D_k)$ declares the parameters of message m. Let $\mathcal{A}$ denote a countably infinite set of process names which are used as recursion variables in recursive specifications. This set can be split into two disjoint subsets $\mathcal{A}_P$ and $\mathcal{A}_N$, and is ranged over by $\mathcal{A}_p$ and $\mathcal{A}_n$ respectively. Moreover, let $X \in \mathcal{A}_p$ and $Z \in \mathcal{A}_n$.

Restricted Broadcast Process Theory (*RBPT*) [6] provides a two-level syntax to define a set of pro-

cesses deployed at a node, also called protocols, and a set of MANETs composed of singleton nodes:

$$P \quad ::= \quad 0 \mid \alpha.P \mid P + P \mid [w]P, P \mid \mathcal{A}_p(\widehat{w})$$
$$N \quad ::= \quad 0 \mid [\![P]\!]_\ell \mid N \parallel N \mid (\nu\ell)N \mid \nabla_M(N) \mid \partial_M(N)$$

A protocol can be a deadlock, modeled by 0. $\alpha.P$ is a process that performs action α and then behaves as process P. The action α can be a send action $m(\widehat{w})!$ or a receive action $m(\widehat{w})?$ where parameters of the receive action can be only (free) variables. The process $P_1 + P_2$ behaves non-deterministically as process P_1 or P_2. The guarded command $[w]P_1, P_2$ defines process behavior based on the data term w of sort $Bool$; if it evaluates to true in the data semantics model, the protocol behaves as P_1, and otherwise as P_2. A process is declared by $\mathcal{A}_p(\langle x : D\rangle) \stackrel{def}{=} P$ where $\mathcal{A}_p$ is a protocol name and x is the variable that appears free in P; for an instantiation $\mathcal{A}_p(\widehat{w})$, it is required that $|\widehat{w}| = |\langle x : D\rangle|$. An occurrence of variable x is bound in P, if it is the parameter of a receive action, or the occurrence is in the scope of a receive action that carries x as a parameter. We restrict to process definitions where each occurrence of $\mathcal{A}_p(\widehat{x})$ in P is in the scope of an action prefix.

Let $\{0, 1\} \in D$, $domain_{req} : I\!P(D)$ and $domain_{rep} : I\!P(D)$ respectively. As a running example, $X_p(x : D) \stackrel{def}{=} req(x)!.X_p(x)$ declares a process that broadcasts a message $req(x)$ recursively, and $X_q \stackrel{def}{=} req(y)?.rep(y)!.X_q$ a process that recursively receives a message $req(u)$ and replies by sending a message $rep(u)$. A MANET can be composed of several nodes using the parallel composition operator, where each node is provided with a unique address ($\ell \neq ?$) and deploys a protocol, and nodes communicate via restricted broadcast. For instance, the network process $[\![X_p(0)]\!]_A \parallel [\![X_q]\!]_B$ specifies a MANET composed of two nodes with logical addresses A and B deploying processes $X_p(0)$ and X_q, respectively. The address of a node can be hidden from an external observer using the restriction operator. For example, in $(\nu A)[\![X_p(0)]\!]_A \parallel [\![X_q]\!]_B$ the activities of node A are hidden from the external observer, and only the activities performed at B can be observed. The abstraction operator $\nabla_M(N)$ restricts the communications of N (with other MANETs) to messages not included in $M \subseteq Msg$: the transmission of messages $m \in M$ are hidden from the external observer. The encapsulation operator $\partial_M(N)$ disallows communications of N on messages in $M \subseteq Msg$, and consequently $\partial_M(N)$ cannot communicate with other MANETs by receiving these messages.

In the following section the syntax of MANETs is extended with new terms, to obtain the class of what we call computed network terms. As the semantics of *RBPT* is subsumed by the one of *CNT*, we postpone an exposition on the formal semantics of *RBPT* until Section 5.

4. Computed Network Theory

As mentioned before, to give the axioms of the equational theory *RBPT*, we use an extension of *RBPT*, called *Computed Network Theory* (*CNT*). This process theory exploits network restrictions, which define a set of topologies; the behavior of process terms is computed with regard to such network restrictions.

CNT extends *RBPT* with new terms called computed networks. A computed network term $C\eta.t$ denotes a network whose behavior, with respect to the set of topologies defined by the network restriction C, is performing the action η and then behaving as t. Parallel composition and restriction are defined over computed networks in the same way as *RBPT* terms. Moreover, *CNT* extends *RBPT* with new operators;

choice ($+$), *left merge* ($\between$), *communication merge* ($|$) and *recursion* $rec\mathcal{A}_n.t$:

$$t ::= 0 \mid [\![P]\!]_\ell \mid C\eta.t \mid t+t \mid t \parallel t \mid t \between t \mid t|t \mid (\nu\ell)t \mid \nabla_M(t) \mid \partial_M(t) \mid \mathcal{A}_n \mid rec\mathcal{A}_n.t$$

where η can be $m(\widehat{u})!\{\ell\}$ or $m(\widehat{u})?$, and C is a network restriction. The choice operator $+$ defines a non-deterministic choice between *CNT* terms, and parallel composition defines computed networks communicating via restricted broadcast. The restriction operator $(\nu)\ell$ conceals the address of a node with logical address ℓ from the external observer as before. The abstraction operator $\nabla_M(t)$ restricts the communications of t to messages not included in M and the encapsulation operator $\partial_M(t)$ disallows communications of network t on messages in M as before. In left merge $\between$, the left operand is succeed to perform the initial action. In the communication merge operator $|$, both operands are succeed to be synchronized on their initial actions. The network name $\mathcal{A}_n$ denotes a specific computed network. The recursion operator $rec\mathcal{A}_n.t$ represents a solution of the equation $\mathcal{A}_n = t$. This solution is unique if $\mathcal{A}_n$ is guarded in t. As far as unguarded recursions are concerned, following the approach of *CCS* and *ACP*, we consider the solution from the set of solutions that has the least set of transitions. In Section 7, we will explain about the guardedness criterion.

The occurrence of an address ℓ is bound in a *CNT* term t by the restriction operator, $(\nu\ell)t$. Bound addresses can be α-converted, meaning that $(\nu\ell)t$ equals $(\nu\ell')t[\ell'/\ell]$ if t does not contain ℓ' as a free address. We define functions $fl(t)$ and $bl(t)$ to denote sets of free and bound addresses in a computed network term t, respectively. An occurrence of a network name $\mathcal{A}_n$ is bound in a *CNT* term t if it occurs in a subterm of the form $rec\mathcal{A}_n.t'$. A computed network term is called closed if its sets of free names and of free variables are empty. We will use P, Q to range over protocols, s, t, u, v to range over *CNT* terms, and $\mathcal{N}, \mathcal{M}$ to range over terms representing processes, i.e. closed *CNT* terms. We use $\equiv$ to denote syntactical equivalence between two terms (protocols and *CNT* terms) and $\equiv_\alpha$ to denote two terms are α-convertable to each other. α-conversion may include renaming of bound addresses, variables and names.

By abuse of notation, we use $t\{t'/\mathcal{A}_n\}$ for expressing replacement of a term t' for every free occurrence of name $\mathcal{A}_n$ in t, if necessary renaming bound names in t in order to ensure that no free occurrence of a name in t' becomes bound in $t\{t'/\mathcal{A}_n\}$.

5. Operational Semantics of *CNT*

A *specification* consists of three parts; data, protocol and network specifications. We can define the static semantics of a specification as in [18], which describes the static requirements under which a specification is defined correctly. For instance, each object should be declared once, and process names, variables, and networks cannot be mixed up. A specification is called *well-formed* if it is statically semantically correct, and its data part has no empty sort and Booleans are defined. For any well-formed specification, the algebraic semantics of its data part must be defined and its equations must constitute a confluent and terminating term rewriting system.

The behavior of protocols and networks is defined using structural operational semantics. The operational semantics of *CNT* is given at two levels (similar to the syntax), in terms of the operational semantics of protocols and of computed network processes.

Given some data model $\mathbb{D}$ and a protocol, the operational rules of Table 1 induce a labeled transition system, in which the transitions are of the form $P \xrightarrow{\alpha} P'$ with α of the form $\{m(\widehat{u})?$ or $m(\widehat{u})!\}$, for some

sequence $\widehat{u}$ of closed data terms. They are standard operational rules for basic process algebras. The Pre_1 and Pre_2 rules indicate execution of receive and send actions. After a process gets a message (of the same type it was ready to receive), values of message parameters are replaced by the corresponding parameters. $Choice$ specifies the choice operator; its symmetric version also holds, but is omitted here. Usually, ad hoc network protocols set timers to perform an action. We can use non-deterministic choice to model time-outs and alternative behaviors of a protocols against external events. A guarded process behaves as P_1 if the guard condition evaluates to true ($Then$), otherwise it behaves as P_2 ($Else$). Process invocation is specified in the standard fashion in Inv, where $P\{\widehat{u}/\widehat{x}\}$ is obtained from P by replacing all occurrences of variables x_i by u_i.

Table 1. Semantics of protocols

$$\frac{}{m(\widehat{x})?.P \xrightarrow{m(\widehat{u})?} P\{\widehat{u}/\widehat{x}\}} : Pre_1 \qquad \frac{}{m(\widehat{u})!.P \xrightarrow{m(\widehat{u})!} P} : Pre_2$$

$$\frac{P\{\widehat{u}/\widehat{x}\} \xrightarrow{\alpha} P'}{\mathcal{A}_p(\widehat{u}) \xrightarrow{\alpha} P'} : Inv,\ \mathcal{A}_p(\langle x:D\rangle) \stackrel{def}{=} P \qquad \frac{P_1 \xrightarrow{\alpha} P_1'}{P_1 + P_2 \xrightarrow{\alpha} P_1'} : Choice$$

$$\frac{P_1 \xrightarrow{\alpha} P_1'}{[u]P_1, P_2 \xrightarrow{\alpha} P_1'} : Then,\ \mathbb{D} \vdash u = T \qquad \frac{P_2 \xrightarrow{\alpha} P_2'}{[u]P_1, P_2 \xrightarrow{\alpha} P_2'} : Else,\ \mathbb{D} \vdash u = F$$

Generally the behavior of a computed network is defined in terms of a set of topologies; a transition, in which a set of nodes participate in a communication, is possible for all topologies in which the receiving nodes are connected to the sending node. Therefore in the operational semantics it is defined for each state which transitions are possible for which sets of topologies (out of all possible topologies). Network restrictions are used to define the set of underlying topologies for each transition.

Given some data model $\mathbb{D}$ and a computed network, the operational rules in Table 2 induce a constrained labeled transition system of transitions $\mathcal{N} \xrightarrow{\eta}_C \mathcal{N}'$, where C is a network restriction defining a set of topologies under which this transition is possible, and η is a send/receive action or τ action. The operational rules of computed networks are shown in Table 2. The symmetric counterparts of rules $Choice'$, Bro, $Sync_2$ and Par have been omitted. Let $hide(C, \ell)$ denote $\{\ell_1 \rightsquigarrow \ell_2 \in C[?/\ell] \mid \ell_2 \neq\ ?\}$ and $Obj(\eta)$ return the message type of η if $\eta \neq \tau$, otherwise τ. The $\delta_M(\eta)$ is defined such that it returns η if $Obj(\eta) \notin M$, otherwise τ. Moreover, let $\eta[\ell'/\ell]$ denote η with all occurrences of ℓ replaced by ℓ'.

$Inter_1$ denotes that a single node can perform the send actions of a protocol at this node under any valid topology, and its network address is appended to this action. $Inter_2$ denotes a single node performing a receive action, under the restriction that the node must be connected to some sender (denoted by ?). Pre' indicates execution of a prefix action. $Choice'$ defines that a computed network can behave non-deterministically. Exe indicates that if a transition is possible for C, then it is also possible for any more restrictive C'. $Recv$ allows to group together nodes that are ready to receive the same message. Bro indicates the actual synchronization in local broadcast among a transmitter and receivers. This transition is valid for all topologies in which the transmitter is connected (not necessarily bidirectionally) to the receivers, which is captured by $C_1 \cup C_2[\ell/?]$. The communication results in a transition labeled with $m(\widehat{u})!\{\ell\}$, so the message $m(\widehat{u})!$ remains visible to be received by other computed networks.

As the communication merge operator defines a successful synchronization between two computed networks, its behavior is defined by $Sync_1$ and $Sync_2$, indicating synchronization on a receive action

(sent by the context) or a communication. $LExe$ defines that in a term composed by the left merge, the left computed network is succeed to perform the initial action, and then the resulting term proceeds as in parallel composition. Par defines locality for a computed network; an event in a computed network may result from this same event in a sub-network.

Table 2. Semantics of *CNT* terms

$$\frac{P \xrightarrow{m(\widehat{u})!} P'}{[\![P]\!]_\ell \xrightarrow{m(\widehat{u})!\{\ell\}}_{\{\}} [\![P']\!]_\ell} : Inter_1 \qquad \frac{P \xrightarrow{m(\widehat{u})?} P'}{[\![P]\!]_\ell \xrightarrow{m(\widehat{u})?}_{\{?\rightsquigarrow\ell\}} [\![P']\!]_\ell} : Inter_2$$

$$\frac{}{C\eta.\mathcal{N} \xrightarrow{\eta}_C \mathcal{N}} : Pre' \qquad \frac{\mathcal{N}_1 \xrightarrow{\eta}_C \mathcal{N}_1'}{\mathcal{N}_1 + \mathcal{N}_2 \xrightarrow{\eta}_C \mathcal{N}_1'} : Choice' \quad \frac{t\{rec\mathcal{A}_n.t/\mathcal{A}_n\} \xrightarrow{\eta}_C \mathcal{N}}{rec\mathcal{A}_n.t \xrightarrow{\eta}_C \mathcal{N}} : Rec$$

$$\frac{\mathcal{N} \xrightarrow{\eta}_C \mathcal{N}'}{\mathcal{N} \xrightarrow{\eta}_{C'} \mathcal{N}'} : Exe, \quad C \subseteq C' \qquad \frac{\mathcal{N}_1 \xrightarrow{m(\widehat{u})?}_{C_1} \mathcal{N}_1' \;\; \mathcal{N}_2 \xrightarrow{m(\widehat{u})?}_{C_2} \mathcal{N}_2'}{\mathcal{N}_1 \parallel \mathcal{N}_2 \xrightarrow{m(\widehat{u})?}_{C_1 \cup C_2} \mathcal{N}_1' \parallel \mathcal{N}_2'} : Recv$$

$$\frac{\mathcal{N}_1 \xrightarrow{m(\widehat{u})!\{\ell\}}_{C_1} \mathcal{N}_1' \;\; \mathcal{N}_2 \xrightarrow{m(\widehat{u})?}_{C_2} \mathcal{N}_2'}{\mathcal{N}_1 \parallel \mathcal{N}_2 \xrightarrow{m(\widehat{u})!\{\ell\}}_{C_1 \cup C_2[\ell/?]} \mathcal{N}_1' \parallel \mathcal{N}_2'} : Bro \quad \frac{\mathcal{N}_1 \xrightarrow{\eta}_C \mathcal{N}_1'}{\mathcal{N}_1 \parallel \mathcal{N}_2 \xrightarrow{\eta}_C \mathcal{N}_1' \parallel \mathcal{N}_2} : Par$$

$$\frac{\mathcal{N}_1 \xrightarrow{m(\widehat{u})?}_{C_1} \mathcal{N}_1' \;\; \mathcal{N}_2 \xrightarrow{m(\widehat{u})?}_{C_2} \mathcal{N}_2'}{\mathcal{N}_1 \mid \mathcal{N}_2 \xrightarrow{m(\widehat{u})?}_{C_1 \cup C_2} \mathcal{N}_1' \parallel \mathcal{N}_2'} : Sync_1 \quad \frac{\mathcal{N}_1 \xrightarrow{\eta}_C \mathcal{N}_1'}{\mathcal{N}_1 \between \mathcal{N}_2 \xrightarrow{\eta}_C \mathcal{N}_1' \parallel \mathcal{N}_2} : LExe$$

$$\frac{\mathcal{N}_1 \xrightarrow{m(\widehat{u})!\{\ell\}}_{C_1} \mathcal{N}_1' \;\; \mathcal{N}_2 \xrightarrow{m(\widehat{u})?}_{C_2} \mathcal{N}_2'}{\mathcal{N}_1 \mid \mathcal{N}_2 \xrightarrow{m(\widehat{u})!\{\ell\}}_{C_1 \cup C_2[\ell/?]} \mathcal{N}_1' \parallel \mathcal{N}_2'} : Sync_2 \quad \frac{\mathcal{N} \xrightarrow{\eta}_C \mathcal{N}'}{(\nu\ell)\mathcal{N} \xrightarrow{\eta[?/\ell]}_{hide(C,\ell)} (\nu\ell)\mathcal{N}'} : Rest$$

$$\frac{\mathcal{N} \xrightarrow{\eta}_C \mathcal{N}'}{\nabla_M(\mathcal{N}) \xrightarrow{\delta_M(\eta)}_C \nabla_M(\mathcal{N}')} : Abs \quad \frac{\mathcal{N} \xrightarrow{\eta}_C \mathcal{N}'}{\partial_M(\mathcal{N}) \xrightarrow{\eta}_C \partial_M(\mathcal{N}')} : Encap, Obj(\eta) \notin M \vee \eta \text{ is a send action}$$

$Rest$ makes sure that restrictions over invisible addresses are removed and the address of a sender with hidden address is concealed from the external observer by converting its address to ?. By using network restrictions, we can easily define the set of topologies over visible nodes under which such a transition is possible (by removing restrictions imposed on hidden nodes). Abs restricts communications of a computed network to a set of messages not included in M. In other words, if the message type of action η belongs to M, $Obj(\eta) \in M$, the action is converted to τ, otherwise η is performed unchanged. The $Encap$ closes the communications of $\partial_M(\mathcal{N})$ on messages in M: receive actions over message types $m \in M$ are prohibited. Recursion is defined in the standard fashion in Rec.

We consider the running example introduced in Section 3. The transition below results from applications of $Inter_1$, $Inter_2$ and Bro:

$$[\![X_p(0)]\!]_A \parallel [\![X_q]\!]_B \xrightarrow{req(0)!\{A\}}_{\{A \rightsquigarrow B\}} [\![X_p(0)]\!]_A \parallel [\![rep(0)!.X_q]\!]_B$$

In this transition, node A broadcasts a message $req(0)$ and node B receives it, so that the parameter x

is substituted by 0. This transition is possible for topologies in which B is connected to A, i.e. the accompanying network restriction is $\{A \rightsquigarrow B\}$. Another possible transition of $[\![X_p(0)]\!]_A \parallel [\![X_q]\!]_B$, resulting from an application of $Inter_1$ and Par, is:

$$[\![X_p(0)]\!]_A \parallel [\![X_q]\!]_B \xrightarrow{req(0)!\{A\}}_{\{\}} [\![X_p(0)]\!]_A \parallel [\![X_q]\!]_B$$

In this transition, node A sends but B does not participate in communication. This transition is possible for all topologies (for instance B may be connected to A, but it has lost the message), denoted by $\{\}$.

If we hide node A, then the possible transitions when A broadcasts (resulting from $Inter_{1,2}$, $Rest$, Bro or Par) are:

$$(\nu A)[\![X_p(0)]\!]_A \parallel [\![X_q]\!]_B \xrightarrow{req(0)!\{?\}}_{\{?\rightsquigarrow B\}} [\![X_p(0)]\!]_A \parallel [\![rep(0)!.X_q]\!]_B$$
$$(\nu A)[\![X_p(0)]\!]_A \parallel [\![X_q]\!]_B \xrightarrow{req(0)!\{?\}}_{\{\}} [\![X_p(0)]\!]_A \parallel [\![X_q]\!]_B$$

Here the observer cannot see who has performed the send action.

6. Branching Computed Network Bisimulation

We define the notion of computed network bisimilarity between nodes in a constrained labeled transition system, based on the notion of branching bisimilarity [10]. To define our equivalence relation, we introduce the following notations:

- $\Rightarrow$ denotes the reflexive and transitive closure of unobservable actions:
  - $\mathcal{N} \Rightarrow \mathcal{N}$;
  - if $\mathcal{N} \xrightarrow{\tau}_C \mathcal{N}'$ for some arbitrary network restriction C and $\mathcal{N}' \Rightarrow \mathcal{N}''$, then $\mathcal{N} \Rightarrow \mathcal{N}''$.
- $\xrightarrow{\langle\eta\rangle}_C$ denotes that either $\xrightarrow{\eta}_C$, or η is of the form $m(\widehat{u})!\{?\}$ and $\xrightarrow{\eta[\ell/?]}_{C[\ell/?]}$.

Definition 6.1. A binary relation $\mathcal{R}$ on computed network terms is a branching computed network simulation, if $\mathcal{N}_1 \mathcal{R} \mathcal{N}_2$ implies whenever $\mathcal{N}_1 \xrightarrow{\eta}_C \mathcal{N}_1'$:

- η is of the form $m(\widehat{u})?$ or τ, and $\mathcal{N}_1' \mathcal{R} \mathcal{N}_2$;
- or there are $\mathcal{N}_2'$ and $\mathcal{N}_2''$ such that $\mathcal{N}_2 \Rightarrow \mathcal{N}_2'' \xrightarrow{\langle\eta\rangle}_C \mathcal{N}_2'$, where $\mathcal{N}_1 \mathcal{R} \mathcal{N}_2''$ and $\mathcal{N}_1' \mathcal{R} \mathcal{N}_2'$.

$\mathcal{R}$ is a branching computed network bisimulation if $\mathcal{R}$ and $\mathcal{R}^{-1}$ are branching computed network simulations. Computed networks $\mathcal{N}_1$ and $\mathcal{N}_2$ are branching computed network bisimilar, written $\mathcal{N}_1 \simeq_b \mathcal{N}_2$, if $\mathcal{N}_1 \mathcal{R} \mathcal{N}_2$ for some branching computed network bisimulation relation $\mathcal{R}$.

Theorem 6.1. Branching computed network bisimilarity is an equivalence.

The proof of above Theorem is given in Appendix A. Computed network bisimilarity is not a congruence with respect to the choice, left merge and communication merge operators. For example the computed networks $\{\}req(u)?.\{\}rep(u)!.0$ and $\{\}rep(u)!.0$ are branching computed network bisimilar, but their congruence is not preserved when they are added with $\{\}req(u)!.0$, since the former can only sends $rep(u)$ while the latter can also sends $req(u)$. To obtain a congruence relation, we need to add a root condition.

Definition 6.2. Two computed networks $\mathcal{N}_1$ and $\mathcal{N}_2$ are *rooted branching computed network bisimilar*, written $\mathcal{N}_1 \simeq_{rb} \mathcal{N}_2$,

- if $\mathcal{N}_1 \xrightarrow{\eta}_C \mathcal{N}_1'$, then there is an $\mathcal{N}_2'$ such that $\mathcal{N}_2 \xrightarrow{\langle\eta\rangle}_C \mathcal{N}_2'$, and $\mathcal{N}_1' \simeq_b \mathcal{N}_2'$;
- if $\mathcal{N}_2 \xrightarrow{\eta}_C \mathcal{N}_2'$, then there is an $\mathcal{N}_1'$ such that $\mathcal{N}_1 \xrightarrow{\langle\eta\rangle}_C \mathcal{N}_1'$, and $\mathcal{N}_1' \simeq_b \mathcal{N}_2'$.

Corollary 6.1. Rooted branching computed network bisimilarity is an equivalence.

Theorem 6.2. Rooted branching computed network bisimilarity is a congruence on *CNT* terms.

The corollary is an immediate result of Theorem 6.1 and Definition 6.2. The proof of Theorem 6.2 is given in Appendix B.

We will also exploit strong bisimilarity [25, 18] to reason about protocol processes:

Definition 6.3. Two protocol processes P_1 and P_2 are strong bisimilar, notation $P_1 \simeq P_2$, iff there is a strong bisimulation relation R over protocol processes such that

- $(P_1, P_2) \in R$,
- for each pair $(P_1', P_2') \in R$:
  - $P_1' \xrightarrow{\alpha} P_1'' \Rightarrow \exists P_2'' \cdot P_2' \xrightarrow{\alpha} P_2''$ and $(P_1'', P_2'') \in R$,
  - $P_2' \xrightarrow{\alpha} P_2'' \Rightarrow \exists P_1'' \cdot P_1' \xrightarrow{\alpha} P_1''$ and $(P_1'', P_2'') \in R$.

The following proposition, relating strong bisimilarity to rooted branching computed network bisimilarity, is straightforward to prove by application of $Inter_{1,2}$ (see Appendix C).

Proposition 6.1. Let P_1 and P_2 be two protocol processes, such that $P_1 \simeq P_2$. Then $[\![P_1]\!]_\ell \simeq_{rb} [\![P_2]\!]_\ell$, where ℓ is an arbitrary location.

7. Protocol and Computed Network Axiomatizations

Table 3 provides standard axioms for closed protocols [17]. Axioms Pr_{0-3} are standard for the choice operator. Axioms $Pr_{4,5}$ define the guarded command behavior regarding condition $[u]$. It can be proved easily that the axiomatization given in Table 3 is sound for the term algebra of protocols modulo strong bisimilarity (cf. [8]), and thus by Proposition 6.1 modulo $\simeq_{rb}$.

Table 3. Axiomatization of protocol terms

$0 + P = P$	Pr_0	$P_1 + P_2 = P_2 + P_1$	Pr_1
$P_1 + (P_2 + Q) = (P_1 + P_2) + Q$	Pr_2	$P + P = P$	Pr_3
$[u]P_1, P_2 = P_1,\ \mathbb{D} \vdash u = T$	Pr_4	$[u]P_1, P_2 = P_2,\ \mathbb{D} \vdash u = F$	Pr_5
$\mathcal{A}_p(\widehat{u}) = P\{\widehat{u}/\widehat{x}\},\ \ \mathcal{A}_p(\langle x : D\rangle) \stackrel{def}{=} P$			Pr_6

Table 4. Axiomatization of *CNT* terms

$P_1 = P_2 \Rightarrow [\![P_1]\!]_\ell = [\![P_2]\!]_\ell$	P_0	$[\![0]\!]_\ell = 0$	P_1
$[\![m(\widehat{u})!.P]\!]_\ell = \{\}m(\widehat{u})!\{\ell\}.[\![P]\!]_\ell$	P_2	$[\![m(\widehat{u})?.P]\!]_\ell = \{? \rightsquigarrow \ell\}m(\widehat{u})?.[\![P]\!]_\ell$	P_3
$[\![m(\widehat{y})?.P]\!]_\ell = \sum_{\widehat{u}\in domain_m}[\![m(\widehat{u})?.P\{\widehat{u}/\widehat{y}\}]\!]_\ell$	P_4	$[\![P_1 + P_2]\!]_\ell = [\![P_1]\!]_\ell + [\![P_2]\!]_\ell$	P_5

$\mathcal{N} + \mathcal{N} = \mathcal{N}$	Cho_1	$\mathcal{N}_1 \parallel \mathcal{N}_2 = \mathcal{N}_1 \between \mathcal{N}_2 + \mathcal{N}_2 \between \mathcal{N}_1 + \mathcal{N}_1 \mid \mathcal{N}_2$	Br
$\mathcal{N}_1 + \mathcal{N}_2 = \mathcal{N}_2 + \mathcal{N}_1$	Cho_2	$C\eta.\mathcal{N}_1 \between \mathcal{N}_2 = C\eta.(\mathcal{N}_1 \parallel \mathcal{N}_2)$	LEx_1
$\mathcal{N}_1 + (\mathcal{N}_2 + \mathcal{N}_3) = (\mathcal{N}_1 + \mathcal{N}_2) + \mathcal{N}_3$	Cho_3	$(\mathcal{N}_1 + \mathcal{N}_2) \between \mathcal{N}_3 = \mathcal{N}_1 \between \mathcal{N}_3 + \mathcal{N}_2 \between \mathcal{N}_3$	LEx_2
$\mathcal{N} + 0 = \mathcal{N}$	Cho_4	$0 \between \mathcal{N} = 0$	LEx_3

$0 = (\nu\ell)0$	$Dead$	$C\eta.(C'm(\widehat{u})?.\mathcal{N} + \mathcal{N}) = C\eta.\mathcal{N}$	T_1
$C_1\eta.\mathcal{N} + C_2\eta.\mathcal{N} = C_1\eta.\mathcal{N}, (C_1 \subseteq C_2)$	Con	$C\eta.(C'\tau.(\mathcal{N}_1 + \mathcal{N}_2) + \mathcal{N}_2) = C\eta.(\mathcal{N}_1 + \mathcal{N}_2)$	T_2
$\mathcal{N}_1 \mid \mathcal{N}_2 = \mathcal{N}_2 \mid \mathcal{N}_1$	S_1	$0 \mid \mathcal{N} = 0$	S_3
$(\mathcal{N}_1 + \mathcal{N}_2) \mid \mathcal{N}_3 = \mathcal{N}_1 \mid \mathcal{N}_3 + \mathcal{N}_2 \mid \mathcal{N}_3$	S_2	$C\tau.\mathcal{N}_1 \mid \mathcal{N}_2 = 0$	S_4

$Cm(\widehat{u})!\{?\}.\mathcal{N} + C[\ell/?]m(\widehat{u})!\{\ell\}.\mathcal{N} = C[\ell/?]m(\widehat{u})!\{\ell\}.\mathcal{N}$ $\quad Obs$

$C_1m(\widehat{u})!\{\ell\}.\mathcal{N}_1 \mid C_2m(\widehat{u})?.\mathcal{N}_2 = C_1 \cup C_2[\ell/?]m(\widehat{u})!\{\ell\}.(\mathcal{N}_1 \parallel \mathcal{N}_2)$	$Sync_1$
$C_1m(\widehat{u}_1)!\{\ell\}.\mathcal{N}_1 \mid C_2n(\widehat{u}_2)?.\mathcal{N}_2 = 0 \quad (m \neq n \vee \widehat{u}_1 \neq \widehat{u}_2)$	$Sync_2$
$C_1m(\widehat{u})?.\mathcal{N}_1 \mid C_2m(\widehat{u})?.\mathcal{N}_2 = C_1 \cup C_2m(\widehat{u})?.(\mathcal{N}_1 \parallel \mathcal{N}_2)$	$Sync_3$
$C_1m(\widehat{u}_1)?.\mathcal{N}_1 \mid C_2n(\widehat{u}_2)?.\mathcal{N}_2 = 0 \quad (m \neq n \vee \widehat{u}_1 \neq \widehat{u}_2)$	$Sync_4$
$C_1m(\widehat{u}_1)!.\mathcal{N}_1\{\ell_1\} \mid C_2n(\widehat{u}_2)!\{\ell_2\}.\mathcal{N}_2 = 0$	$Sync_5$

$(\nu\ell)(\mathcal{N}_1 + \mathcal{N}_2) = (\nu\ell)\mathcal{N}_1 + (\nu\ell)\mathcal{N}_2$	Res_1
$(\nu\ell)\eta.\mathcal{N} = hide(C, \ell)\eta.(\nu\ell)\mathcal{N} \quad (\eta = m(\widehat{u})!\{\ell'\} \wedge \ell \neq \ell') \vee \eta = \tau$	Res_2
$(\nu\ell)\,Cm(\widehat{u})!\{\ell\}.\mathcal{N} = hide(C, \ell)m(\widehat{u})!\{?\}.(\nu\ell)\mathcal{N}$	Res_3
$(\nu\ell)\,Cm(\widehat{u})?.\mathcal{N} = hide(C, \ell)m(\widehat{u})?.(\nu\ell)\mathcal{N}$	Res_4

$\partial_M(C\eta.\mathcal{N}) = C\eta.\partial_M(\mathcal{N}), Obj(\eta) \notin M \vee \eta$ is a send $\quad Ecp_1$

$\nabla_M(C\eta.\mathcal{N}) = C\delta_M(\eta).\nabla_M(\mathcal{N})$	Abs_1	$\partial_M(Cm(\widehat{u})?.\mathcal{N}) = 0, m \in M$	Ecp_2
$\nabla_M(\mathcal{N}_1 + \mathcal{N}_2) = \nabla_M(\mathcal{N}_1) + \nabla_M(\mathcal{N}_2)$	Abs_2	$\partial_M(\mathcal{N}_1 + \mathcal{N}_2) = \partial_M(\mathcal{N}_1) + \partial_M(\mathcal{N}_2)$	Ecp_3
$\nabla_M(0) = 0$	Abs_3	$\partial_M(0) = 0$	Ecp_4

$rec\mathcal{A}_n.t = t\{rec\mathcal{A}_n.t/\mathcal{A}_n\}$	$Unfold$
$s = t\{s/\mathcal{A}_n\} \Rightarrow s = rec\mathcal{A}_n.t, \mathcal{A}_n$ guarded in t	$Fold$
$rec\mathcal{A}_n.(\mathcal{A}_n + t) = rec\mathcal{A}_n.t$	Ung
$rec\mathcal{A}_n.(C\tau.(C'\tau.t' + t) + s) = rec\mathcal{A}_n.(C\tau.(t' + t) + s), \mathcal{A}_n$ unguarded in t'	$WUng_1$
$rec\mathcal{A}_n.(C\eta_\tau.(\mathcal{A}_n + t) + s) = rec\mathcal{A}_n.(C\eta_\tau.(t + s) + s), \eta_\tau \in \{m(\widehat{u})?, \tau\}$	$WUng_2$
$\nabla_M(rec\mathcal{A}_n.t) = rec\mathcal{A}_n.\nabla_M(t), \mathcal{A}_n$ is serial in t	Hid

We proceed to present an axiomatization for the process theory *CNT*, modulo rooted branching computed network bisimilarity. The axioms are given in Table 4. P_0 says that two equal protocols, when deployed at the same node, result in equal networks. Axioms P_{1-5} relate the behavior of a protocol deployed at a node to a computed network term. In P_4, summation $\sum$ is used to denote a choice over a finite set of data elements, in this case $domain_m$; summation over an empty set denotes 0. *Dead* explains that hiding an address in a deadlock computed network has no effect. Con expresses that if the same behavior happens under two different sets of topologies, and one set is included in the other set, then from the view point of an external observer, the behavior occurs for the superset of topologies. Obs expresses when a send from a node with a hidden address has no effect and can be equated to any send from a node with a visible address. Cho_{1-4} define idempotency, commutativity, associativity and unit element for the choice operator. The parallel composition of two computed networks is defined in an interleaving semantics, as in the process algebra *ACP* [3], by the axiom Br; in a network composed of two computed networks $\mathcal{N}_1$ and $\mathcal{N}_2$, each network may perform a local action (since they are not in the range of each other or due to noise in the environment they do not succeed to synchronize), or they may communicate via local broadcast. LEx_{1-3} define axioms for the left merge: the left operand can perform an action (LEx_1), the choice operator can be distributed over the left merge (LEx_2), and when the left operand cannot do any action, then the left merge results into a deadlock (LEx_3). S_1 and S_2 define commutativity and distributivity of choice over the communication merge operator, respectively. S_3 defines that when an argument in a communication merge composition is a deadlock, then the result of the composition is a deadlock. When an argument in a communication merge composition can only performs τ, then the result is deadlock as explained in S_4 (because the τ action originates from a wireless communication which occupies the common communication medium, and therefore it cannot be ignored as in [3]). $Sync_{1-5}$ define synchronization between two computed network terms. Generally speaking, two terms can be synchronized if they send/receive the same message with the same parameter values. T_1 and T_2 express when a receive and τ action can be removed respectively. Res_1 defines distribution of restriction over the choice operator. Res_{2-4} express the effect of the restriction operator: network restrictions over hidden addresses are removed. In Res_2, restriction has no effect on τ or send actions from visible addresses, except for removing restrictions over hidden addresses. In Res_3, the address of a hidden sender is converted to $?$. Abs_1 explains the effect of the abstraction operator on a computed network: each action η is converted to τ by $\delta_M(\eta)$ if $Obj(\eta) \in M$, otherwise η is unchanged. Abs_2 defines distribution of abstraction over the choice operator. Ecp_1 and Ecp_2 explain that an encapsulation operator parameterized by a set of messages M prohibits a computed network to communicate with other networks by receiving a message $m \in M$. Ecp_3 defines distribution of abstraction over the choice operator. Abs_3 and Ecp_4 defines the effect of abstraction and encapsulation operators on deadlock.

We define a guardedness criterion for network names to ensure that a network name $\mathcal{A}_n$ specified by $\mathcal{A}_n = t$ has a unique solution, denoted by $rec\mathcal{A}_n.t$. A free occurrence of a network name $\mathcal{A}_n$ in t is called *guarded* if this occurrence is in the scope of an action prefix operator (not τ prefix) and not in the scope of an abstraction operator [1]; in other words, there is a subterm $C\eta.t'$ in t such that $\eta \neq \tau$, and $\mathcal{A}_n$ occurs in t'. $\mathcal{A}_n$ is *(un)guarded* in t if (not) every free occurrence of $\mathcal{A}_n$ in t is guarded. A *CNT* term t is *guarded* if for every subterm $rec\mathcal{A}_n.t'$, $\mathcal{A}_n$ is guarded in t'. This guardedness criterion ensures that any guarded recursive term has a unique solution. To understand why $C\tau$ does not ensure that a recursion has one solution, consider the following example: $recZ_n.(C\tau.Z_n)$ has solutions like $C\tau.0$ and $C\tau.C'req(u)!.0$, while they are not rooted branching computed network bisimilar.

$Unfold$ and $Fold$ express existence and uniqueness of a solution for the equation $\mathcal{A}_n = t$, which

correspond to Milner's standard axioms, and the *Recursive Definition Principle* (*RDP*) and *Recursive Specification Principle* (*RSP*) in *ACP*. $Unfold$ states that each recursive operator has a solution (whether it is guarded or not), while $Fold$ states that each guarded recursive operator has at most one solution. So $[\![X_p(0)]\!]_A$ and $[\![X_q]\!]_B$, by application of Pr_6, $P_{0,2-4}$ and $Unfold$, can be converted to

$$[\![X_p(0)]\!]_A = \{\}req(0)!\{A\}.[\![X_p(0)]\!]_A \Rightarrow [\![X_p(0)]\!]_A = recX_n.(\{\}req(0)!\{A\}.X_n)$$

$$[\![X_q]\!]_B = \sum_{i=0,1}\{? \rightsquigarrow B\}req(i)?.\{\}rep(i)!\{B\}.[\![X_q]\!]_B \Rightarrow$$
$$[\![X_q]\!]_B = recX_m.\sum_{i=0,1}(\{? \rightsquigarrow B\}req(i)?.\{\}rep(i)!\{B\}.X_m)$$

It should be noted that protocol names with recursive definitions, when deployed at a node, result in computed network terms with recursive behaviors. Protocol names with definitions in which protocol names prefixed by actions, can be easily turned into a guarded recursion operator by application of axioms Pr_{4-6}, P_{1-5} and $Unfold$ (see above example). However, a recursive term in the scope of an abstraction would become unguarded, as we will explain below. Axioms Ung, $WUng_1$ and $WUng_2$ make it possible to turn each unguarded recursion into a guarded one.

A complete axiomatization of finite-state behaviors in the context of branching bisimilarity has given in [8]; four axioms for removal of unguardedness were provided, as shown below:

$$\mu X(X + E) = \mu XE \qquad R_3$$
$$\mu X(\tau(\tau E + F) + G) = \mu X(\tau(E + F) + G), \text{provided } X \text{ is unguarded in } E \qquad R_4$$
$$\mu X(\tau(X + E) + \tau(X + F) + G) = \mu X(\tau(X + E + F) + G) \qquad R_5$$
$$\mu X(\tau(X + E) + F) = \mu X(\tau(E + F) + F) \qquad R_6$$

where E, F, G are meta-variables that range over open terms, X a variable, and μXE represents a solution of the equation $X = E$. Our axioms Ung, $WUng_1$ and $WUng_2$ correspond to the axioms R_3, R_4 and R_6, respectively. Besides axiom $WUng_2$ removes self-loop receive action, as a receiving node is equivalent to an empty computed network. Axiom R_5 is derivable from R_4 and R_6 as shown below:[1]

$$\mu X(\tau(X + E) + \tau(X + F) + G) =^{R_6}$$
$$\mu X(\tau.(\tau(X + F) + G + E) + \tau(X + F) + G) =^{R_4}$$
$$\mu X(\tau.(X + F + G + E) + \tau(X + F) + G) =^{R_6}$$
$$\mu X(\tau.(\tau(X + F + G + E) + F + G) + \tau.(X + F + G + E) + G) =^{R_4}$$
$$\mu X(\tau.(X + F + G + E) + G) =^{R_6}$$
$$\mu X(\tau.(F + G + E) + G) =^{R_6}$$
$$\mu X(\tau.(X + F + E) + G)$$

Axiom Hid expresses that the abstraction operator can be moved inside and outside of a recursion operator, when $\mathcal{A}_n$ is serial in t. The free network name $\mathcal{A}_n$ is *serial* in t, if it does not occur in the scope of parallel, communication merge, left merge, restriction, encapsulation, and abstraction operators in t. This side condition is required to preserve the soundness of axiom as explained in [1] (This axiom

[1] Personal communication with Rob van Glabbeek learned us that he is aware of this, but never wrote it down.

was also considered in [9], which needs a side condition to be sound in the context of *CCS*). It should be noted that the abstraction operator can make a guarded recursion unguarded. Thus by applying axiom Hid, we can move the operator inside the recursion operator and apply its effect, which may result in unguarded recursion. Then by moving it out and applying $WUng_1$ and $WUng_2$, we can convert it to a guarded one. Finally, by applying $Unfold$, we can remove the abstraction operator completely.

Theorem 7.1. The axiomatization is sound for the term algebra $\mathbb{P}(CNT)/\simeq_{rb}$, i.e. for all closed computed network terms $\mathcal{N}_1$ and $\mathcal{N}_2$, if $\mathcal{N}_1 = \mathcal{N}_2$ then $\mathcal{N}_1 \simeq_{rb} \mathcal{N}_2$.

The proof of this theorem is presented in Appendix D. Using the given axioms in Table 4, we can derive sound axioms over *RBPT* terms given in Table 5. Axiom Par_{1-3} explain commutativity, associativity and unit element for parallel composition for the parallel operator as expected. $Res_{5,6}$ explain that number of repeats and the order of the restriction operator have no effect on the behavior of computed network terms. Res_7 explains scope extrusion of the restriction operator.

Table 5. Sound axioms over *RBPT* terms

$N_1 \parallel N_2 = N_2 \parallel N_1$	Par_1	$(\nu\ell)N = N, \ell \notin fl(N)$	Res_5
$N_1 \parallel (N_2 \parallel N_3) = (N_1 \parallel N_2) \parallel N_3$	Par_2	$(\nu\ell_1)(\nu\ell_2)N = (\nu\ell_2)(\nu\ell_1)N$	Res_6
$N_1 \parallel 0 = 0$	Par_3	$(\nu\ell)N_1 \parallel N_2 = (\nu\ell)(N_1 \parallel N_2), \ell \notin fl(N_2)$	Res_7

We apply the axioms in Table 4 to the running example. The following equations indicate that the behavior of $[\![X_p(0)]\!]_A \parallel [\![X_q]\!]_B$, when its communication to receive req and rep is restricted, is: A can broadcast a message but B does not participate, or A can broadcast a message and B receives it for a set of topologies in which B is connected to A.

$$\begin{aligned}
&\mathcal{M}_1 \equiv [\![X_p(0)]\!]_A = \{\}req(0)!\{A\}.recX_n.(\{\}req(0)!.X_n) \equiv \{\}req(0)!\{A\}.\mathcal{M}_1\\
&\mathcal{M}_2 \equiv [\![X_q]\!]_B = \textstyle\sum_{i=0,1}\{? \rightsquigarrow B\}req(i)?.\{\}rep(i)!\{B\}.\\
&\qquad\qquad recX_m.\textstyle\sum_{i=0,1}(\{? \rightsquigarrow B\}req(i)?.\{\}rep(i)!\{B\}.\mathcal{A}_m)\\
&\qquad\qquad \equiv \textstyle\sum_{i=0,1}\{? \rightsquigarrow B\}req(i)?.\{\}rep(i)!.\mathcal{M}_2
\end{aligned}$$

$$\begin{aligned}
&\partial_{\{req,rep\}}(\mathcal{M}_1 \parallel \mathcal{M}_2) = \partial_{\{req,rep\}}(\mathcal{M}_1 \between \mathcal{M}_2 + \mathcal{M}_2 \between \mathcal{M}_1 + \mathcal{M}_1 \mid \mathcal{M}_2)\\
&\quad = \partial_{\{req,rep\}}(\{\}req(0)!\{A\}.\mathcal{M}_1 \between \mathcal{M}_2 + \textstyle\sum_{i=0,1}\{? \rightsquigarrow B\}req(i)?.\{\}rep(i)!.\mathcal{M}_2 \between \mathcal{M}_1\\
&\qquad + \{\}req(0)!\{A\}.\mathcal{M}_1 \mid \textstyle\sum_{i=0,1}\{? \rightsquigarrow B\}req(i)?.\{\}rep(i)!.\mathcal{M}_2)\\
&\quad = \{\}req(0)!\{A\}.\partial_{\{req,rep\}}(\mathcal{M}_1 \parallel \mathcal{M}_2) + \{A \rightsquigarrow B\}req(0)!\{A\}.\partial_{\{req,rep\}}(\mathcal{M}_1 \parallel \{\}rep(i)!.\mathcal{M}_2).
\end{aligned}$$

Let C be a hidden node that relays req messages that it receives: $X_m \stackrel{def}{=} req(x)?.req(x)!.X_m$. Consider a network consisting of nodes A, B and C where the address of C is hidden. For such a network, we can derive $\partial_{\{req,rep\}}([\![X_p(0)]\!]_A \parallel [\![X_q]\!]_B) = \partial_{\{req,rep\}}([\![X_p(0)]\!]_A \parallel [\![X_q]\!]_B \parallel (\nu C)[\![X_m]\!]_C)$, since both are a solution of the following recursive term:

$$recZ_n.(\{\}req(0)!\{A\}.Z_n + \{A \rightsquigarrow B\}req(0)!\{A\}.recZ_{n_0}.(\{\}req(0)!\{A\}.Z_{n_0} + \{\}rep(0)!\{B\}.Z_n)).$$

Thus a node that relays messages has no effect on the behavior of the network.

Protocols X_p, X_q and X_m can be a part of a simple routing protocol, each specifying the behavior of a node as the initiator (to find a route to a specific destination), destination, and middle node (which relays messages from the initiator toward the destination) respectively. We complete the definitions of these protocols as given below. To increase the readability of code, we write $\mathcal{A}_{p_0} \stackrel{def}{=} m(u)?.\mathcal{A}_{p_1}$ instead of $\mathcal{A}_{p_0} \stackrel{def}{=} m(x)?.[x == u]\mathcal{A}_{p_1}, \mathcal{A}_{p_0}$. Process X_{p_0} keeps the address of the next hop on a route to the destination by the variable $next$. When it does not know any path to the destination, it broadcasts req recursively until it finds a route by receiving $rep(x)$ from the next hop with an address x. Otherwise when X_{p_0} knows the address of the next hop, it sends data through the next hop to the destination. However, it may receive an $error$ message from the next hop, which indicates it cannot be used as a router to the destination (due to link breakage or some failure at the node). In this case the process will set $next$ to the unknown address ?.

$$
\begin{aligned}
&X_{p_0}(next : Loc, addr : Loc)_{\textcircled{0}} \stackrel{def}{=}\\
&\quad [next! =?]data(next)!.X_{p_0}(next, addr) + error(next)?.X_{p_0}(?, addr)\\
&\qquad , req!._{\textcircled{1}} X_{p_1}(addr)\\
&X_{p_1}(addr : Loc) \stackrel{def}{=} rep(x)?.X_{p_0}(x, addr) + req!.X_{p_1}(addr)\\
&\\
&X_{m_0}(next : Loc, addr : Loc)_{\textcircled{0}} \stackrel{def}{=}\\
&\quad [next! =?](data(addr)?._{\textcircled{1}} data(next)!.X_{m_0}(next, addr)+\\
&\qquad\qquad error(next)?._{\textcircled{2}} error(addr)!.X_{m_0}(?, addr)+\\
&\qquad\qquad error(addr)!.X_{m_0}(?, addr))\\
&\qquad , req?._{\textcircled{3}} req!.X_{m_0}(next, addr) + rep(x)?._{\textcircled{4}} rep(addr)!.X_{m_0}(x, addr)\\
&\\
&X_{q_0}(addr : Loc)_{\textcircled{0}} \stackrel{def}{=}\\
&\quad req?._{\textcircled{1}} rep(addr)!.X_{q_0}(addr) + data(addr)?.X_{q_0}(addr) + error(addr)!.X_{q_0}(addr)
\end{aligned}
$$

The behavior of a middle node, X_{m_0}, when it does not know a path to the destination is the same as before (i.e. relaying req, and also rep messages), but when it receives a rep message, it will keep this address to reply to req messages in the future. When a middle node finds a route to the destination, it also relays $data$ and $error$ message. In this case, it also non-deterministically sends $error$ to model link breakage between itself and next hop. The process of a destination node, X_{q_0}, replies to req messages by sending rep as before, and receives $data$. It also sends $error$ in occurrence of a failure at the node.

We are going to examine if this simple protocol, when its processes are deployed on nodes in a MANET, i.e. $[\![X_{p_0}(?, A)]\!]_A \parallel (\nu C)[\![X_{m_0}(?, C)]\!]_C \parallel [\![X_{q_0}(B)]\!]_B$, then data messages are correctly routed from the initiator to the destination. The specification of the MANET in which data messages are routed from the source address A to the destination address B is:

$$
\begin{aligned}
&\{\}\tau.recZ_{n_0}.(\{\}data(B)!\{A\}.Z_{n_0} + \{\}data(?)!\{A\}.Z_{n_0}+\\
&\qquad\qquad \{\}data(?)!\{A\}.recZ_{n_1}.(\{\}data(?)!\{A\}.Z_{n_1} + \{\}data(B)!\{?\}.Z_{n_0}))
\end{aligned}
$$

which indicates (after finding the route modeled by the initial τ action) A sends its data to B directly, or A sends its data to an unknown location, which will send data to B. We can use our axioms to examine if $\nabla_{\{req,rep,error\}}(\partial_{\{rep,rep,error,data\}}([\![X_{p_0}(?, A)]\!]_A \parallel (\nu C)[\![X_{m_0}(?, C)]\!]_C \parallel [\![X_{q_0}(B)]\!]_B))$ is a solution for Z_{n_0}. Let $\chi(s_A, next_A, s_B, s_C, next_C)$ stand for

$$\partial_{\{rep,rep,error,data\}}([\![P_{s_A}\{next_A/next\}]\!]_A \parallel [\![P_{s_B}]\!]_B \parallel (\nu C)[\![P_{s_C}\{next_C/next\}]\!]_C)$$

where P_i is a subprocess of P after the $\textcircled{i}$ indicator in previous specifications. It can be proved that (see Appendix F):

$$\nabla_{\{req,rep,error\}}(\chi(0,?,0,0,?)) = \nabla_{\{req,rep,error\}}(recZ.\{\}\tau.\chi(0,B,0,0,?) + \{\}\tau.\chi(0,C,0,0,B)) \quad (1)$$

where $\chi(0,B,0,0,?)$ specifies the scenario in which X_{p_0} has a direct route to B at the beginning (since in future it may use another paths) and consequently sends its data directly to B, while $\chi(0,C,0,0,B)$ specifies the scenario in which X_{p_0} has found a route via C to B at the beginning and consequently sends its data via C to B. By application of axioms, it can be proved that the following equations hold:

$$\begin{aligned}&\{\}\tau.\nabla_{\{req,rep,error\}}(\chi(0,B,0,0,?)) = \\ &\quad \{\}\tau.\nabla_{\{req,rep,error\}}(recZ_B.\{\}data(B)!\{A\}.Z_B + \{B \rightsquigarrow A\}\tau.\chi(0,?,0,0,?))\end{aligned} \quad (2)$$

$$\begin{aligned}&\{\}\tau.\nabla_{\{req,rep,error\}}(\chi(0,C,0,0,B)) = \\ &\quad \{\}\tau.\nabla_{\{req,rep,error\}}(recZ_C.\{\}data(?)!\{A\}.Z_C + \{\}\tau.\chi(0,?,0,0,?)+ \\ &\qquad\qquad \{\}data(?)!\{A\}.recZ_{CB}.(\{\}data(?)!\{A\}.Z_{CB} + \{\}data(B)!\{?\}.Z_C)+ \\ &\qquad\qquad \{\}\tau.recZ_e.(\{\}data(?)!\{A\}.Z_e + \{? \rightsquigarrow A\}\tau.\chi(0,?,0,0,?)+ \\ &\qquad\qquad\qquad \{\}\tau.recZ_d.\{\}data(?)!\{A\}.Z_d)).\end{aligned} \quad (3)$$

By application of axioms $WUng_{1-2}$, $UnFold$, Hid and equations 1-3, it can be proved that:

$$\begin{aligned}&\nabla_M(\chi(0,?,0,0,?)) = \{\}\tau.(\\ &\quad \{\}data(B)!\{A\}.\nabla_M(\chi(0,?,0,0,?)) + \{\}data(?)!\{A\}.\nabla_M(\chi(0,?,0,0,?))+ \\ &\quad \{\}data(?)!\{A\}.recZ_{CB}.(\{\}data(?)!\{A\}.Z_{CB} + \{\}data(B)!\{?\}.\nabla_M(\chi(0,?,0,0,?)))+ \\ &\quad \{\}\tau.recZ_d.\{\}data(?)!\{A\}.Z_d).\end{aligned} \quad (4)$$

which is very similar to the specification of a MANET. However, there is a loop defined by Z_d in which A recursively sends data to some unknown node (i.e. C), but that node does not route data to B. This scenario happens when A routes data through a middle node. If the link between the middle node and B breaks down, then it broadcasts $error$ message. If A loses this error message, it never finds out about the invalidity of its next hop and continues to forward its data to the next hop. One of the solutions is to assign a timer to each route, which becomes invalid when the timer times out. However, if we change the definition of X_{m_0} such that it retransmits $error$ even after its next hop is unknown, then Z_d is removed from the above equation. We can also prove such a property by adding more middle nodes while their addresses are hidden. The restriction operator helps to specify behaviors for a number of unknown locations.

8. Completeness of the Axiomatization for Finite-state Behaviors

We prove that the axiomatization in Table 3 and 4 is ground-complete for *CNT* terms with finite-state models, modulo rooted branching computed network bisimilarity. Following the approach of [1], to restrict to *CNT* terms with finite-state constrained labeled transition systems, we provide a syntactical restriction for recursive terms $rec\mathcal{A}_n.t$. We consider so-called *finite-state Computed Network Theory* (*CNT*$_f$), which is obtained by restricting the closed computed networks in the *CNT* syntax: every recursive term $rec\mathcal{A}_n.t$ must be *essentially finite-state*, which means that its bound network names do not

occur in the scope of parallel, communication merge, left merge, restriction, encapsulation, and abstraction operators in t. Recall that $\mathcal{A}_n$ is serial in t when $rec\mathcal{A}_n.t$ is essentially finite-state. For instance, $recZ_n.(\{\}req(0)!\{A\}.0 \parallel \{?\rightsquigarrow B\}req(x)?.Z_n)$ is not essentially finite-state; it produces an infinite-state transition system, since at each recursive call, a new parallel operator is generated.

It is trivial to see that each finite-state process can be described by an essentially finite-state recursive term. Conversely we can show that every CNT_f term has finitely many states in the transition system generated by the operational rules. See Appendix E for the proof.

Proposition 8.1. Let $\mathcal{N}$ be a closed CNT_f term such that every subterm $rec\mathcal{A}_n.t$ is essentially finite-state. Given a data model $\mathbb{D}$ with finite data domains, the transition system for $\mathcal{N}$ generated by the operational rules has only finitely many states.

Theorem 8.1. Given a data model $\mathbb{D}$ with finite data domains, the axiomatization is ground-complete for the term algebra $\mathbb{P}(CNT_f)/\simeq_{rb}$, i.e. for all closed finite-state computed network terms $\mathcal{N}_1$ and $\mathcal{N}_2$, $\mathcal{N}_1 \simeq_{rb} \mathcal{N}_2$ implies $\mathcal{N}_1 = \mathcal{N}_2$.

The proof of the above theorem is presented in Appendix G.

9. Related Work

Related calculi to ours are CBS#, CWS, CMAN, CMN, ω-calculus and SCWN [24, 22, 11, 12, 20, 26, 13]. In all these related approaches, the only equations between networks were defined by using structural congruence. None of these papers provides a complete axiomatization for their algebra of MANETs. We shortly go through these calculi, with a focus on their approach in modeling topology and mobility, and on their purpose of verification.

CBS# [24], an extension of CBS, provides a framework for specification and security analysis of communication protocols for MANETs. In this approach, the mobility is modeled implicitly in the semantics. The operational semantics is parameterized by a set of connectivity graphs, each imposing a set of connections between nodes in a network. Each transition of a MANET is parameterized by a connectivity graph. In other words, the connectivity graph defines the behavior of a network at each step, while in our approach the behavior of a network defines the set of topologies under which such a behavior can occur. Consequently we merge all transitions, and their corresponding topologies leading to the same state, into a transition subscripted by a network restriction. Thus our approach results in a more compact labeled transition system.

CWS [22] (Calculus for Wireless Systems) is a channel-based algebra for modeling MAC-layer protocols, for which interferences are an essential aspect. In this approach the physical characteristics of nodes such as their physical location and transmission ranges are considered, while locations of nodes are static. CMN [20] (Calculus of Mobile ad hoc Networks), inspired by CWS to model MANETs above the MAC-layer, concerns modeling the mobility of nodes explicitly in the semantics. To this aim, for each node a physical location is specified and the underlying topology is derived by a function d, which takes two locations and computes their distance. If the distance is smaller than a pre-defined value, nodes at those locations are connected. Mobility is modeled by changing the location of the node to an arbitrary location, which may lead to state space explosion if locations are drawn from a real coordinate system.

CMAN [11] (Calculus of Mobile Ad hoc Networks) provides an approach for modeling of MANETs where for each node, a connectivity set (its neighbors) is specified. Mobility of a node is modeled explicitly in semantics by manipulation of the connectivity set of all effected nodes, by adding/removing this node from/to their connectivity set. In a more recent work [12], CMAN has been extended by a static location binding operator to limit the arbitrary mobility of nodes in the scope of the operator, so that a MANET can be verified for a specific mobility scenario.

The ω-calculus [26], a conservative extension of the π-calculus, provides an approach to specify MANETs in the same vein as CMAN, but models connectivity information, called process interface, at the specification level by a group concept; a group is a maximal clique in a connectivity graph, and two nodes can communicate if they belong to the same group. Node mobility is captured through the dynamic creation of new groups and dynamically changing process interfaces, using appropriate rules in the semantics. By defining a mobility invariant which constrains node mobilities, one can derive the model of a MANET and verify it against a mobility scenario. In contrast, using our approach, one can verify the model of a MANET against different mobility scenarios by defining predicates over network restrictions.

Recently in [13], a simple process calculus with broadcast operator was extended by realistic mobility models in an orthogonal way, so-called SCWN (a Simple Calculus for Wireless Networks). This approach, which can be understood as a generalization of CWS, is appropriate to verify properties under a specific mobility model, in contrast to the arbitrary mobility model. To this aim, the specification of a node is equipped with a mobility function which determines the movement trajectories of a node over time and consequently its neighbors; the semantics incorporates a notion of global time passing and is parameterized by a mobility model which manipulates the mobility function of a node. This method is based on computations of the transmission range of a sender using physical locations of nodes to derive the real underlying topology. This approach suffers from state explosion because of its real-time delay transitions, which may be resolved by using a discrete time delay (as proposed by its inventors). In

Table 6. Comparison between related algebras

	Node specification	Connectivity	Mobility
CBS#	$\ell[p, s]$	$-$	implicit
CWS	$n[p]^c_{l,r}$	derived by $d(l, l')$	$-$
CMN	$n[p]^c_{l,r}$	derived by $d(l, l')$	explicit
CMAN	$[p]^\sigma_\ell$	σ	explicit
ω-calculus	$p : g$	g	explicit
RBPT	$[\![p]\!]_\ell$	$-$	implicit
SCWN	$\ell[p]^T_f$	derived by $area_n(f(t))$	explicit

Table 6, we have compared our core algebra, *RBPT*, with the related ones in terms of node specification, how connectivity information is specified or derived, and how mobility is modeled, where n refers to the name of the node, ℓ to the logical address, l to the physical location, r to the transmission range, σ to the connectivity set, and g to the connected groups of a node. Finally, f denotes the mobility function, which defines the location of a node at time t, and T is a timeout when the mobility function is updated.

A behavioral congruence is based on an external observer that observes a system through a limited set

of observables (called barbs). A MANET consists of nodes that each have a (physical or logical network) location/address and a transmission range. Different observables are defined based on how the address of nodes, the transmitted values and the transmission range of nodes are considered [14]. However, it is not obvious to decide what are adequate observables for mobile and local wireless broadcasting calculi [22, 14]. In CBS#, two networks are distinguished in terms of their capability to store data terms in a network location: the pairs of data terms and network locations are the observables. In CMN, two networks are distinguished in terms of transmissions over a channel name and the possible receivers on that channel: message transmission, communication channel and the range of a transmitter are taken into the account. The behavioral congruence in CMAN is defined in terms of locality of transmissions. Thus two process with the same number of transmissions deployed at the same network location are equivalent irrespective of the values transmitted: this framework is intended for the detection of intruders which cause some extra transmissions from some locations. In ω-calculus two MANETs are distinguished in terms of their capability to send or receive data values from a set of groups. The behavioral congruence is defined by a bisimulation relation and does not offer an explicit definition of observables, but since it is a conservative extension of the π-calculus at least contains the observables of the π-calculus, i.e. channel names [14]. In SCWN, two MANETs, each under a mobility model, are distinguished in terms of their capability to send and receive data terms over an area in a time interval. In *RBPT*, two networks are distinguished in terms of their capability to transmit data from a location for a set of topologies. Our behavioral congruence is defined by a bisimulation relation, but at least the locations of nodes and connectivities of nodes (i.e. range) are observables. In Table 7, we used our notation for send and receive actions as a unifying notation to show that an equivalence relation holds in each framework under the corresponding behavioral congruence, denoted by $=$, between the basic networks consisting of one node.

Table 7. Comparison of behavioral congruence relations

	Observables	Equivalent MANETs	Distinguished MANETs
CBS#	data, location	$\ell[m(x)?.0, s] = 0$	$\ell[m(x)?.store.0, s] \neq 0$, $\ell[p, s] \neq \ell'[p, s]$
CWS	-	-	-
CMN	channel, range	$n[p]_{l,r} = n[p]_{l',r}$, $n[m(x)?.0]_{l,r} = 0$	-
CMAN	location	$[m(u)!.0]_\ell^{\sigma_1} = [n(u)!.0]_\ell^{\sigma_2}$, $[m(u)?.0]_\ell^\sigma = 0$	$[p]_\ell^\sigma \neq [p]_{\ell'}^\sigma$
ω-calculus	channel	$p : g_1 = p : g_2$	$m(x)?.0 : g \neq 0$
RBPT	range, location	$[\![m(x)?.0]\!]_\ell = 0$, $(\nu\ell')[\![p]\!]_{\ell'} = (\nu\ell)[\![p]\!]_\ell$	$[\![p]\!]_\ell \neq [\![p]\!]_{\ell'}$ $(\nu\ell')[\![p]\!]_{\ell'} = [\![p]\!]_\ell$

In [14], different observables for process calculi for mobile and wireless broadcasting systems in the context of weak barbed congruences are discussed. The observables consider location, data and channel names (which is applicable to channel-based process calculi). It is proved that behavioral congruence based on location observables is included in the one based on data, which coincides with the one based on channel and location observables. It should be noted that in our framework for properties, when locations are not needed to be considered, the restriction operator can be used. These properties are more

related to the overall behavior of a MANET. In this case, our behavioral congruence is based on range observables and it coincides with the one in CMN (since broadcast is the only communication channel in *RBPT*). On the other hand, for properties related to some specific nodes (identified by their locations) in a MANET, like finding a route between two nodes, the observability of locality is helpful.

10. Conclusion and Future Work

We introduced Restricted Broadcast Process Theory (*RBPT*) to specify and verify MANETs, taking into account local broadcast communication and mobility. We modeled mobility implicitly in semantics, by allowing arbitrary switches between topologies in each state. Our approach to model mobility and formalize behavior of a network with respect to the set of topologies, modeled by network constraints, leads to a compact labeled transition. Moreover, by transferring mobility concepts to the semantics, our process algebra provides a natural way to model MANETs, because topologies are not a part of the network specification.

To axiomatize *RBPT* terms, we extended it with new terms and operators, to obtain Computed Network Theory (*CNT*). The behavior of *CNT* terms is computed with respect to a set of topologies, specified by a network restriction. We gave an operational semantics, and defined an equivalence notion between computed networks, called rooted branching computed network bisimilarity. We provided a sound axiomatization for *CNT* terms modulo rooted branching computed network bisimilarity. Our axiomatization is ground-complete for *CNT* terms with a finite-state model; we have classified such terms by imposing restrictions on recursive terms.

We are going to use mCRL2 [15] to write an interpreter based on our axiomatization, following the approach of [19], which converts the specifications written in our algebra to its corresponding constrained labeled transition system. We can analyze a MANET by model checking using the modal μ-calculus. We would like to extend our framework with proof techniques to reason about protocols with an infinite data domain. Following [12], we intend to add an operator which abstracts away from the movements we are not interested in. Finally, want to extend our calculus with stochastic concepts to evaluate MANET protocols. Since the communication media is not reliable (and so MAC-layer protocols), performance evaluation of protocols above the MAC layer is important to measure their quality. The main challenge is how to incorporate the rate of delays from the upper network layer (e.g. protocol) with ones in the lower layer network (e.g. the MAC-layer).

References

[1] Baeten, J., Bravetti, M.: A Ground-Complete Axiomatization of Finite State Processes in Process Algebra, *Proc. 16th Conference on Concurrency Theory (CONCUR)*, LNCS 3653, pages 248–262, Springer, 2005.

[2] Basten, T.: Branching Bisimilarity is an Equivalence Indeed!, *Information Processing Letters*, **58**(3), 1996, 141–147.

[3] Bergstra, J., Klop, J.: Process Algebra for Synchronous Communication, *Information and Control*, **60**(1-3), 1984, 109–137.

[4] Ehrich, H., Loeckx, J., Wolf, M.: *Specification of Abstract Data Types*, John Wiley, 1996.

[5] Ene, C., Muntean, T.: A Broadcast-based Calculus for Communicating Systems, *Proc. 15th Parallel & Distributed Processing Symposium (IPDPS)*, page 149, IEEE, 2001.

[6] Ghassemi, F., Fokkink, W., Movaghar, A.: Restricted Broadcast Process Theory, *Proc. 6th Conference on Software Engineering and Formal Methods (SEFM)*, pages 345–354, IEEE, 2008.

[7] Ghassemi, F., Fokkink, W., Movaghar, A.: Equational Reasoning on Ad Hoc Networks, *Proc. 3rd Conference on Fundamentals of Software Engineering (FSEN)*, LNCS 5961, pages 113-128, Springer, 2009.

[8] van Glabbeek, R.: A Complete Axiomatization for Branching Bisimulation Congruence of Finite-State Behaviours, *Proc. 18th Symposium on Mathematical Foundations of Computer Science (MFCS)*, LNCS 711, pages 473–484, Springer, 1993.

[9] van Glabbeek, R.: Notes on the Methodology of CCS and CSP, *Theoretical Computer Science*, **177**(2), 1997, 329–349.

[10] van Glabbeek, R., Weijland, W.: Branching Time and Abstraction in Bisimulation Semantics, *Journal of the ACM*, **43**(3), 1996, 555–600.

[11] Godskesen, J.: A Calculus for Mobile Ad Hoc Networks, *Proc. 9th Conference on Coordination Models and Languages (COORDINATION)*, LNCS 4467, pages 132–150, Springer, 2007.

[12] Godskesen, J.: A Calculus for Mobile Ad-hoc Networks with Static Location Binding, *Electronic Notes in Theoretical Computer Science*, **242**(1), 2009, 161–183.

[13] Godskesen, J., Nanz, S.: Mobility Models and Behavioural Equivalence for Wireless Networks, *Proc. 11th Conference on Coordination Models and Languages (COORDINATION)*, LNCS 5521, pages 106–122, Springer, 2009.

[14] Godskesen, J. C.: Observables for Mobile and Wireless Broadcasting Systems, *Proc. 12th Conference on Coordination Models and Languages (COORDINATION)*, LNCS 6116, pages 1–15, Springer, 2010.

[15] Groote, J., Mathijssen, A., Reniers, M. A., Usenko, Y. S., van Weerdenburg, M.: The Formal Specification Language mCRL2, *Proc. Methods for Modelling Software Systems (MMOSS)*, vol. 06351 of *Dagstuhl Seminar Proceedings*, 2006.

[16] Groote, J., Ponse, A.: μCRL: A Base for Analysing Processes with Data, *Proc. 3rd Workshop on Concurrency and Compositionality*, pages 125-130, GMD-Studien Nr. 191, 1991.

[17] Groote, J., Ponse, A.: Proof Theory for μCRL: A Language for Processes with Data, *Semantics of Specification Languages*, Workshops in Computing, pages 232–251, Springer, 1993.

[18] Groote, J., Ponse, A.: Syntax and Semantics of μ-CRL, *Algebra of Communicating Processes*, Workshops in Computing, pages 26–62, Springer, 1995.

[19] Hojjat, H., Mousavi, M. R., Sirjani, M.: A Framework for Performance Evaluation and Functional Verification in Stochastic Process Algebras, *Proc. ACM Symposium on Applied Computing (ACM SAC)*, pages 339-346, ACM, 2008.

[20] Merro, M.: An Observational Theory for Mobile Ad Hoc Networks, *Information and Computation*, **207**(2), 2009, 194–208.

[21] Merro, M., Sibilio, E.: A Timed Calculus for Wireless Systems, *Proc. 3rd Conference on Fundamentals of Software Engineering (FSEN)*, LNCS 5961, pages 228-243, Springer, 2010.

[22] Mezzetti, N., Sangiorgi, D.: Towards a Calculus For Wireless Systems, *Electronic Notes in Theoretical Computer Science*, **158**, 2006, 331–353.

[23] Milner, R.: A Complete Axiomatisation for Observational Congruence of Finite-State Behaviors, *Information and Computation*, **81**(2), 1989, 227–247.

[24] Nanz, S., Hankin, C.: A Framework for Security Analysis of Mobile Wireless Networks, *Theoretical Computer Science*, **367**(1), 2006, 203–227.

[25] Park, D.: Concurrency and Automata on Infinite Sequences, *Proc. 5th GI-Conference on Theoretical Computer Science*, LNCS 104, pages 167–183, Springer, 1981.

[26] Singh, A., Ramakrishnan, C. R., Smolka, S. A.: A Process Calculus for Mobile Ad Hoc Networks, *Science of Computer Programming*, **75**(6), 2010, 440–469.

A. Branching Computed Network Bisimilarity is an Equivalence

To prove that branching computed network bisimilarity is an equivalence, we exploit semi-branching computed network bisimilarity, following [2]. In the next definition, $\mathcal{N} \xrightarrow{(\eta)}_C \mathcal{N}'$ denotes either $\mathcal{N} \xrightarrow{\eta}_C \mathcal{N}'$, or $\eta \in \{m(\widehat{u})?, \tau\}$ and $\mathcal{N} = \mathcal{N}'$.

Definition A.1. A binary relation $\mathcal{R}$ on computed network terms is a semi-branching computed network simulation, if $\mathcal{N}_1 \mathcal{R} \mathcal{N}_2$ implies whenever $\mathcal{N}_1 \xrightarrow{\eta}_C \mathcal{N}_1'$:

- there are $\mathcal{N}_2'$ and $\mathcal{N}_2''$ such that $\mathcal{N}_2 \Rightarrow \mathcal{N}_2'' \xrightarrow{(\langle\eta\rangle)}_C \mathcal{N}_2'$, where $\mathcal{N}_1 \mathcal{R} \mathcal{N}_2''$ and $\mathcal{N}_1' \mathcal{R} \mathcal{N}_2'$.

$\mathcal{R}$ is a semi-branching computed network bisimulation if $\mathcal{R}$ and $\mathcal{R}^{-1}$ are semi-branching computed network simulations. Computed networks $\mathcal{N}_1$ and $\mathcal{N}_2$ are semi-branching computed network bisimilar if $\mathcal{N}_1 \mathcal{R} \mathcal{N}_2$, for some semi-branching computed network bisimulation relation $\mathcal{R}$.

Lemma A.1. Let $\mathcal{N}_1$ and $\mathcal{N}_2$ be computed network terms, and $\mathcal{R}$ a semi-branching computed network bisimulation such that $\mathcal{N}_1 \mathcal{R} \mathcal{N}_2$.

- If $\mathcal{N}_1 \Rightarrow \mathcal{N}_1'$ then $\exists \mathcal{N}_2' \cdot \mathcal{N}_2 \Rightarrow \mathcal{N}_2' \wedge \mathcal{N}_1' \mathcal{R} \mathcal{N}_2'$
- If $\mathcal{N}_2 \Rightarrow \mathcal{N}_2'$ then $\exists \mathcal{N}_1' \cdot \mathcal{N}_1 \Rightarrow \mathcal{N}_1' \wedge \mathcal{N}_1' \mathcal{R} \mathcal{N}_2'$

Proof:
We only give the proof of the first property. The second property can be proved in a similar fashion. The proof is by induction on the number of $\Rightarrow$ steps from $\mathcal{N}_1$ to $\mathcal{N}_1'$:

- Base: Assume that the number of steps equals zero. Then $\mathcal{N}_1$ and $\mathcal{N}_1'$ must be equal. Since $\mathcal{N}_1 \mathcal{R} \mathcal{N}_2$ and $\mathcal{N}_2 \Rightarrow \mathcal{N}_2$, the property is satisfied.
- Induction step: Assume $\mathcal{N}_1 \Rightarrow \mathcal{N}_1'$ in n steps, for some $n \geq 1$. Then there is an $\mathcal{N}_1''$ such that $\mathcal{N}_1 \Rightarrow \mathcal{N}_1''$ in $n-1 \Rightarrow$ steps, and $\mathcal{N}_1'' \xrightarrow{\tau}_C \mathcal{N}_1'$. By the induction hypothesis, there exists an $\mathcal{N}_2''$ such that $\mathcal{N}_2 \Rightarrow \mathcal{N}_2''$ and $\mathcal{N}_1'' \mathcal{R} \mathcal{N}_2''$. Since $\mathcal{N}_1'' \xrightarrow{\tau}_C \mathcal{N}_1'$ and $\mathcal{R}$ is a semi-branching computed network bisimulation, there are two cases to consider:
  - there is an $\mathcal{N}_2'$ such that $\mathcal{N}_2'' \Rightarrow \mathcal{N}_2'$, $\mathcal{N}_1'' \mathcal{R} \mathcal{N}_2'$, and $\mathcal{N}_1' \mathcal{R} \mathcal{N}_2'$. So $\mathcal{N}_2 \Rightarrow \mathcal{N}_2'$ such that $\mathcal{N}_1' \mathcal{R} \mathcal{N}_2'$.
  - or there are $\mathcal{N}_2'''$ and $\mathcal{N}_2'$ such that $\mathcal{N}_2'' \Rightarrow \mathcal{N}_2''' \xrightarrow{\tau}_C \mathcal{N}_2'$, where $\mathcal{N}_1'' \mathcal{R} \mathcal{N}_2'''$ and $\mathcal{N}_1' \mathcal{R} \mathcal{N}_2'$. By definition, $\mathcal{N}_2''' \xrightarrow{\tau}_C \mathcal{N}_2'$ yields $\mathcal{N}_2''' \Rightarrow \mathcal{N}_2'$. Consequently $\mathcal{N}_2 \Rightarrow \mathcal{N}_2'$ such that $\mathcal{N}_1' \mathcal{R} \mathcal{N}_2'$.

□

Proposition A.1. The relation composition of two semi-branching computed network bisimulations is again a semi-branching computed network bisimulation.

Proof:
Let $\mathcal{R}_1$ and $\mathcal{R}_2$ be semi-branching computed network bisimulations with $\mathcal{N}_1\mathcal{R}_1\mathcal{N}_2$ and $\mathcal{N}_2\mathcal{R}_2\mathcal{N}_3$. Let $\mathcal{N}_1 \xrightarrow{\eta}_C \mathcal{N}_1'$. It must be shown that

$$\exists \mathcal{N}_3', \mathcal{N}_3'' : \mathcal{N}_3 \Rightarrow \mathcal{N}_3'' \xrightarrow{(\langle\eta\rangle)} \mathcal{N}_3' \wedge \mathcal{N}_1\mathcal{R}_1 \circ \mathcal{R}_2\mathcal{N}_3'' \wedge \mathcal{N}_1'\mathcal{R}_1 \circ \mathcal{R}_2\mathcal{N}_3'$$

Since $\mathcal{N}_1\mathcal{R}_1\mathcal{N}_2$, there exist $\mathcal{N}_2'$, $\mathcal{N}_2''$ such that $\mathcal{N}_2 \Rightarrow \mathcal{N}_2'' \xrightarrow{(\langle\eta\rangle)}_C \mathcal{N}_2'$, $\mathcal{N}_1\mathcal{R}_1\mathcal{N}_2''$ and $\mathcal{N}_1'\mathcal{R}_1\mathcal{N}_2'$. Since $\mathcal{N}_2\mathcal{R}_2\mathcal{N}_3$ and $\mathcal{N}_2 \Rightarrow \mathcal{N}_2''$, Lemma A.1 yields that there is an $\mathcal{N}_3''$ such that $\mathcal{N}_3 \Rightarrow \mathcal{N}_3''$ and $\mathcal{N}_2''\mathcal{R}_2\mathcal{N}_3''$. Two cases can be distinguished:

- $\eta \in \{m(\widehat{u})?, \tau\}$ and $\mathcal{N}_2'' = \mathcal{N}_2'$. It follows immediately that $\mathcal{N}_3 \Rightarrow \mathcal{N}_3'' \xrightarrow{(\langle\eta\rangle)}_C \mathcal{N}_3''$, $\mathcal{N}_1\mathcal{R}_1 \circ \mathcal{R}_2\mathcal{N}_3''$ and $\mathcal{N}_1'\mathcal{R}_1 \circ \mathcal{R}_2\mathcal{N}_3''$.

- Assume $\mathcal{N}_2'' \xrightarrow{\langle\eta\rangle}_C \mathcal{N}_2'$. Since $\mathcal{N}_2''\mathcal{R}_2\mathcal{N}_3''$ and $\mathcal{R}_2$ is a semi-branching computed network bisimulation, there are $\mathcal{N}_3'''$ and $\mathcal{N}_3'$ such that $\mathcal{N}_3'' \Rightarrow \mathcal{N}_3''' \xrightarrow{(\langle\eta\rangle)}_C \mathcal{N}_3'$, $\mathcal{N}_2''\mathcal{R}_2\mathcal{N}_3'''$ and $\mathcal{N}_2'\mathcal{R}_2\mathcal{N}_3'$. Since $\mathcal{N}_3 \Rightarrow \mathcal{N}_3''$, we have $\mathcal{N}_3 \Rightarrow \mathcal{N}_3''' \xrightarrow{(\langle\eta\rangle)}_C \mathcal{N}_3'$. Furthermore, $\mathcal{N}_1\mathcal{R}_1 \circ \mathcal{R}_2\mathcal{N}_3'''$ and $\mathcal{N}_1'\mathcal{R}_1 \circ \mathcal{R}_2\mathcal{N}_3'$.

□

Corollary A.1. Semi-branching computed network bisimilarity is an equivalence relation.

Proposition A.2. Each largest semi-branching computed network bisimulation is a branching computed network bisimulation.

Proof:
Suppose $\mathcal{R}$ is the largest semi-branching computed network bisimulation for some given constrained labeled transition systems. Let $\mathcal{N}_1\mathcal{R}\mathcal{N}_2$, $\mathcal{N}_2 \Rightarrow \mathcal{N}_2'$, $\mathcal{N}_1\mathcal{R}\mathcal{N}_2'$ and $\mathcal{N}_1'\mathcal{R}\mathcal{N}_2'$. We show that $\mathcal{R}' = \mathcal{R} \cup \{(\mathcal{N}_1', \mathcal{N}_2)\}$ is a semi-branching computed network bisimulation.

1. If $\mathcal{N}_1' \xrightarrow{\eta}_C \mathcal{N}_1''$, then it follows from $(\mathcal{N}_1', \mathcal{N}_2') \in \mathcal{R}$ that there are $\mathcal{N}_2'''$ and $\mathcal{N}_2''$ such that $\mathcal{N}_2' \Rightarrow \mathcal{N}_2''' \xrightarrow{(\langle\eta\rangle)}_C \mathcal{N}_2''$ with $(\mathcal{N}_1', \mathcal{N}_2'''), (\mathcal{N}_1'', \mathcal{N}_2'') \in \mathcal{R}$. And $\mathcal{N}_2 \Rightarrow \mathcal{N}_2'$ yields $\mathcal{N}_2 \Rightarrow \mathcal{N}_2''' \xrightarrow{(\langle\eta\rangle)}_C \mathcal{N}_2''$.

2. If $\mathcal{N}_2 \xrightarrow{\eta}_C \mathcal{N}_2''$, then it follows from $(\mathcal{N}_1, \mathcal{N}_2) \in \mathcal{R}$ that there are $\mathcal{N}_1'''$ and $\mathcal{N}_1''$ such that $\mathcal{N}_1 \Rightarrow \mathcal{N}_1''' \xrightarrow{(\langle\eta\rangle)}_C \mathcal{N}_1''$ with $(\mathcal{N}_1''', \mathcal{N}_2), (\mathcal{N}_1'', \mathcal{N}_2'') \in \mathcal{R}$. Since $(\mathcal{N}_1, \mathcal{N}_2') \in \mathcal{R}$ and $\mathcal{N}_1 \Rightarrow \mathcal{N}_1'''$, by Lemma A.1, there is an $\mathcal{N}_2'''$ such that $\mathcal{N}_2' \Rightarrow \mathcal{N}_2'''$ and $(\mathcal{N}_1''', \mathcal{N}_2''') \in \mathcal{R}$. Since $\mathcal{N}_1''' \xrightarrow{(\langle\eta\rangle)}_C \mathcal{N}_1''$, there are $\mathcal{N}_2^{**}$ and $\mathcal{N}_2^{*}$ such that $\mathcal{N}_2''' \Rightarrow \mathcal{N}_2^{**} \xrightarrow{(\langle\eta\rangle)}_C \mathcal{N}_2^{*}$ with $(\mathcal{N}_1''', \mathcal{N}_2^{**}), (\mathcal{N}_1'', \mathcal{N}_2^{*}) \in \mathcal{R}$. Since $\mathcal{N}_2' \Rightarrow \mathcal{N}_2'''$ and $\mathcal{N}_2''' \Rightarrow \mathcal{N}_2^{**}$, we have $\mathcal{N}_2' \Rightarrow \mathcal{N}_2^{**}$. By assumption, $(\mathcal{N}_1', \mathcal{N}_2') \in \mathcal{R}$, so by Lemma A.1 there is an $\mathcal{N}_1^{**}$ such that $\mathcal{N}_1' \Rightarrow \mathcal{N}_1^{**}$ and $(\mathcal{N}_1^{**}, \mathcal{N}_2^{**}) \in \mathcal{R}$. Since $\mathcal{N}_2^{**} \xrightarrow{(\langle\eta\rangle)}_C \mathcal{N}_2^{*}$, there

are $\mathcal{N}_1^{***}$ and $\mathcal{N}_1^{*}$ such that $\mathcal{N}_1^{**} \Rightarrow \mathcal{N}_1^{***} \xrightarrow{(\langle\eta\rangle)}_C \mathcal{N}_1^{*}$ with $(\mathcal{N}_1^{***}, \mathcal{N}_2^{**}), (\mathcal{N}_1^{*}, \mathcal{N}_2^{*}) \in \mathcal{R}$. And $\mathcal{N}_1' \Rightarrow \mathcal{N}_1^{**}$ yields $\mathcal{N}_1' \Rightarrow \mathcal{N}_1^{***} \xrightarrow{(\langle\eta\rangle)}_C \mathcal{N}_1^{*}$.

$$(\mathcal{N}_1^{***}, \mathcal{N}_2^{**}) \in \mathcal{R} \wedge (\mathcal{N}_2^{**}, \mathcal{N}_1''') \in \mathcal{R}^{-1} \wedge (\mathcal{N}_1''', \mathcal{N}_2) \in \mathcal{R}$$
$$\Rightarrow (\mathcal{N}_1^{***}, \mathcal{N}_2) \in \mathcal{R} \circ \mathcal{R}^{-1} \circ \mathcal{R}$$
$$(\mathcal{N}_1^{*}, \mathcal{N}_2^{*}) \in \mathcal{R} \wedge (\mathcal{N}_2^{*}, \mathcal{N}_1'') \in \mathcal{R}^{-1} \wedge (\mathcal{N}_1'', \mathcal{N}_2'') \in \mathcal{R}$$
$$\Rightarrow (\mathcal{N}_1^{*}, \mathcal{N}_2'') \in \mathcal{R} \circ \mathcal{R}^{-1} \circ \mathcal{R}$$

By Proposition A.1, $\mathcal{R} \circ \mathcal{R}^{-1} \circ \mathcal{R}$ is a semi-branching computed network bisimulation. Since $\mathcal{R}$ is the largest semi-branching computed network bisimulation, and clearly $\mathcal{R} \subseteq \mathcal{R} \circ \mathcal{R}^{-1} \circ \mathcal{R}$, we have $\mathcal{R} = \mathcal{R} \circ \mathcal{R}^{-1} \circ \mathcal{R}$. Concluding, $\mathcal{N}_1' \Rightarrow \mathcal{N}_1^{***} \xrightarrow{(\langle\eta\rangle)}_C \mathcal{N}_1^{*}$ with $(\mathcal{N}_1^{***}, \mathcal{N}_2), (\mathcal{N}_1^{*}, \mathcal{N}_2'') \in \mathcal{R}$.

So $\mathcal{R}'$ is a semi-branching computed network bisimulation. Since $\mathcal{R}$ is the largest semi-branching computed network bisimulation, $\mathcal{R}' = \mathcal{R}$.

We will now prove that $\mathcal{R}$ is a branching computed network bisimulation. Let $\mathcal{N}_1 \mathcal{R} \mathcal{N}_2$, and $\mathcal{N}_1 \xrightarrow{\eta}_C \mathcal{N}_1'$. We only consider the case when η is of the form $m(\widehat{u})?$ or τ denoted by η_τ, because for other cases, the transfer condition of Definition 6.1 and Definition A.1 are the same. So there are $\mathcal{N}_2''$ and $\mathcal{N}_2'$ such that $\mathcal{N}_2 \Rightarrow \mathcal{N}_2'' \xrightarrow{(\eta_\tau)}_C \mathcal{N}_2'$ with $\mathcal{N}_1 \mathcal{R} \mathcal{N}_2''$ and $\mathcal{N}_1' \mathcal{R} \mathcal{N}_2'$. Two cases can be distinguished:

1. $\mathcal{N}_2'' = \mathcal{N}_2'$: Since $\mathcal{N}_1 \mathcal{R} \mathcal{N}_2$, $\mathcal{N}_1 \mathcal{R} \mathcal{N}_2'$, and $\mathcal{N}_1' \mathcal{R} \mathcal{N}_2'$, we proved above that $\mathcal{N}_1' \mathcal{R} \mathcal{N}_2$. This agrees with the first case of Definition 6.1.

2. $\mathcal{N}_2'' \neq \mathcal{N}_2'$: This agrees with the second case of Definition 6.1.

Consequently $\mathcal{R}$ is a branching computed network bisimulation. □

Lemma A.2. Let $\mathcal{R}$ be the largest branching computed network bisimulation given for some constrained labeled transition systems. If there exist $\mathcal{N}_1 \xrightarrow{\tau}_{C_0} \mathcal{N}_1^{*} \xrightarrow{\tau}_{C_1} \cdots \xrightarrow{\tau}_{C_{m-1}} \mathcal{N}_m^{*} \xrightarrow{\tau}_{C_m} \mathcal{N}_1'$, where $m \geq 0$, such that $\mathcal{N}_1 \mathcal{R} \mathcal{N}_2$ and $\mathcal{N}_1' \mathcal{R} \mathcal{N}_2$, then $\forall_{1 \leq i \leq m} : \mathcal{N}_i^{*} \mathcal{R} \mathcal{N}_2$.

Proof:
Suppose $\mathcal{R}$ is the largest semi-branching computed network bisimulation for the given constrained labeled transition systems. We show that $\mathcal{R}' = \mathcal{R} \bigcup_{1 \leq i \leq m} \{(\mathcal{N}_i^{*}, \mathcal{N}_2)\}$ is a semi-branching computed network bisimulation. To this aim, it suffices to show that each pair $(\mathcal{N}_i^{*}, \mathcal{N}_2) \in \mathcal{R}'$, $i \leq m$, satisfies the transfer condition of Definition A.1

- If $\mathcal{N}_i^{*} \xrightarrow{\eta}_C \mathcal{N}_i^{*'}$, then $\mathcal{N}_1 \xrightarrow{\tau}_C \mathcal{N}_1^{*} \xrightarrow{\tau}_{C_1} \cdots \xrightarrow{\tau}_{i-1} \mathcal{N}_i^{*} \xrightarrow{\eta}_C \mathcal{N}_i^{*'}$, and since $(\mathcal{N}_1, \mathcal{N}_2) \in \mathcal{R}$, there is a sequence $\mathcal{N}_2 \Rightarrow \mathcal{N}_1^{**} \Rightarrow \cdots \Rightarrow \mathcal{N}_i^{**}$ such that $(\mathcal{N}_1^{*}, \mathcal{N}_1^{**}), \cdots, (\mathcal{N}_i^{*}, \mathcal{N}_i^{**}) \in \mathcal{R}$. It follows from $(\mathcal{N}_i^{*}, \mathcal{N}_i^{**}) \in \mathcal{R}$ that there exist $\mathcal{N}_i^{**''}$ and $\mathcal{N}_i^{**'}$ such that $\mathcal{N}_i^{**} \Rightarrow \mathcal{N}_i^{**''} \xrightarrow{(\langle\eta\rangle)}_C \mathcal{N}_i^{**'}$ with $(\mathcal{N}_i^{*}, \mathcal{N}_i^{**''}), (\mathcal{N}_i^{*'}, \mathcal{N}_i^{**'}) \in \mathcal{R}$. Hence $\mathcal{N}_2 \Rightarrow \mathcal{N}_i^{**''} \xrightarrow{(\langle\eta\rangle)}_C \mathcal{N}_i^{**'}$ with $(\mathcal{N}_i^{*}, \mathcal{N}_i^{**''}), (\mathcal{N}_i^{*'}, \mathcal{N}_i^{**'}) \in \mathcal{R}'$ as required.

- If $\mathcal{N}_2 \xrightarrow{\eta}_C \mathcal{N}_2'$, then it follows from $(\mathcal{N}_1', \mathcal{N}_2) \in \mathcal{R}$ that there exist $\mathcal{N}_1'''$ and $\mathcal{N}_1''$ such that $\mathcal{N}_1' \Rightarrow \mathcal{N}_1''' \xrightarrow{(\langle\eta\rangle)}_C \mathcal{N}_1''$ with $(\mathcal{N}_1''', \mathcal{N}_2), (\mathcal{N}_1'', \mathcal{N}_2') \in \mathcal{R}$. Hence, there is a path $\mathcal{N}_i^{*} \Rightarrow \mathcal{N}_1''' \xrightarrow{(\langle\eta\rangle)}_C \mathcal{N}_1''$ with $(\mathcal{N}_1''', \mathcal{N}_2), (\mathcal{N}_1'', \mathcal{N}_2') \in \mathcal{R}'$.

Thus $\mathcal{R}'$ is a semi-branching computed network bisimulation, and since it is the largest, we have $\mathcal{R} = \mathcal{R}'$. Using Proposition A.2, we conclude the proof. □

Since any branching computed network bisimulation is a semi-branching computed network bisimulation, this yields the following corollary.

Corollary A.2. Two computed network terms are related by a branching computed network bisimulation if and only if they are related by a semi-branching computed network bisimulation.

Corollary A.3. Branching computed network bisimilarity is an equivalence relation.

Corollary A.4. Rooted branching computed network bisimilarity is an equivalence relation.

Proof:
It is easy to show that rooted branching computed network bisimilarity is reflexive and symmetric. To conclude the proof, we show that rooted branching computed network bisimilarity is transitive. Let $\mathcal{N}_1 \simeq_{rb} \mathcal{N}_2$ and $\mathcal{N}_2 \simeq_{rb} \mathcal{N}_3$. Since $\mathcal{N}_1 \simeq_{rb} \mathcal{N}_2$, if $\mathcal{N}_1 \xrightarrow{\eta}_C \mathcal{N}_1'$, then there is an $\mathcal{N}_2'$ such that $\mathcal{N}_2 \xrightarrow{\langle\eta\rangle}_C \mathcal{N}_2'$ and $\mathcal{N}_1' \simeq_b \mathcal{N}_2'$. Since $\mathcal{N}_2 \simeq_{rb} \mathcal{N}_3$, there is an $\mathcal{N}_3'$ such that $\mathcal{N}_3 \xrightarrow{\langle\eta\rangle}_C \mathcal{N}_3'$ and $\mathcal{N}_2' \simeq_b \mathcal{N}_3'$. Since branching computed network bisimilarity is an equivalence, $\mathcal{N}_3 \xrightarrow{\langle\eta\rangle}_C \mathcal{N}_3'$ with $\mathcal{N}_1' \simeq_b \mathcal{N}_3'$. The same argumentation holds when $\mathcal{N}_3 \xrightarrow{\langle\eta\rangle}_C \mathcal{N}_3'$. Consequently the transfer conditions of Definition 6.2 hold and $\mathcal{N}_1 \simeq_{rb} \mathcal{N}_3$.
□

B. Rooted Branching Computed Network Bisimilarity is a Congruence

Theorem B.1. Rooted branching computed network bisimilarity is a congruence with respect to the protocol and computed network operators.

Proof:
We need to prove the following cases:

1. $[\![P_1]\!]_\ell \simeq_{rb} [\![P_2]\!]_\ell$ implies $[\![\alpha.P_1]\!]_\ell \simeq_{rb} [\![\alpha.P_2]\!]_\ell$
2. $[\![P_1]\!]_\ell \simeq_{rb} [\![P_2]\!]_\ell$ and $[\![P_1']\!]_\ell \simeq_{rb} [\![P_2']\!]_\ell$ implies $[\![P_1 + P_1']\!]_\ell \simeq_{rb} [\![P_2 + P_2']\!]_\ell$
3. $[\![P_1]\!]_\ell \simeq_{rb} [\![P_2]\!]_\ell$ and $[\![P_1']\!]_\ell \simeq_{rb} [\![P_2']\!]_\ell$ implies $[\![[u_1 = u_2]P_1, P_1']\!]_\ell \simeq_{rb} [\![[u_1 = u_2]P_2, P_2']\!]_\ell$
4. $\mathcal{N}_1 \simeq_{rb} \mathcal{N}_2$ implies $C\eta.\mathcal{N}_1 \simeq_{rb} C\eta.\mathcal{N}_2$
5. $\mathcal{N}_1 \simeq_{rb} \mathcal{N}_2$ and $\mathcal{N}_1' \simeq_{rb} \mathcal{N}_2'$ implies $\mathcal{N}_1 + \mathcal{N}_1' \simeq_{rb} \mathcal{N}_2 + \mathcal{N}_2'$
6. $\mathcal{N}_1 \simeq_{rb} \mathcal{N}_2$ implies $(\nu\ell)\mathcal{N}_1 \simeq_{rb} (\nu\ell)\mathcal{N}_2$
7. $\mathcal{N}_1 \simeq_{rb} \mathcal{N}_2$ and $\mathcal{N}_1' \simeq_{rb} \mathcal{N}_2'$ implies $\mathcal{N}_1 \parallel \mathcal{N}_1' \simeq_{rb} \mathcal{N}_2 \parallel \mathcal{N}_2'$
8. $\mathcal{N}_1 \simeq_{rb} \mathcal{N}_2$ and $\mathcal{N}_1' \simeq_{rb} \mathcal{N}_2'$ implies $\mathcal{N}_1 \between \mathcal{N}_1' \simeq_{rb} \mathcal{N}_2 \between \mathcal{N}_2'$

9. $\mathcal{N}_1 \simeq_{rb} \mathcal{N}_2$ and $\mathcal{N}_1' \simeq_{rb} \mathcal{N}_2'$ implies $\mathcal{N}_1 \mid \mathcal{N}_1' \simeq_{rb} \mathcal{N}_2 \mid \mathcal{N}_2'$
10. $\mathcal{N}_1 \simeq_{rb} \mathcal{N}_2$ implies $\partial_M(\mathcal{N}_1) \simeq_{rb} \partial_M(\mathcal{N}_2)$
11. $\mathcal{N}_1 \simeq_{rb} \mathcal{N}_2$ implies $\nabla_M(\mathcal{N}_1) \simeq_{rb} \nabla_M(\mathcal{N}_2)$

Clearly, if $\mathcal{N}_1 \simeq_{rb} \mathcal{N}_2$ then $\mathcal{N}_1 \simeq_b \mathcal{N}_2$ witnessed by the following branching computed network bisimulation relation:

$$\begin{aligned}\mathcal{R}' = \{&\mathcal{R} | \mathcal{N}_1 \xrightarrow{\eta}_C \mathcal{N}_1' \Rightarrow \exists \mathcal{N}_2' \cdot \mathcal{N}_2 \xrightarrow{\langle\eta\rangle}_C \mathcal{N}_2' \wedge \mathcal{N}_1' \simeq_b \mathcal{N}_2' \text{ witnessed by } \mathcal{R}\}\\ &\cup \{\mathcal{R} | \mathcal{N}_2 \xrightarrow{\eta}_C \mathcal{N}_2' \Rightarrow \exists \mathcal{N}_1' \cdot \mathcal{N}_1 \xrightarrow{\langle\eta\rangle}_C \mathcal{N}_1' \wedge \mathcal{N}_1' \simeq_b \mathcal{N}_2' \text{ witnessed by } \mathcal{R}\}\\ &\cup \{(\mathcal{N}_1, \mathcal{N}_2)\}.\end{aligned}$$

We prove cases 1, 2, 6, 9 and 10 since the proofs of case 3 and 5 are similar to case 2, case 4 is similar to case 1, cases 7, 8 are similar to case 9, and case 11 is similar to case 10.

Case 1. The first transitions of $[\![\alpha.P_1]\!]_\ell$ and $[\![\alpha.P_2]\!]_\ell$ are the same, and since $[\![P_1]\!]_\ell \simeq_{rb} [\![P_1]\!]_\ell$ then $[\![P_1]\!]_\ell \simeq_b [\![P_1]\!]_\ell$. Thus the transfer conditions of Definition 6.2 hold.

Case 2. Every transition $[\![P_1 + P_1']\!]_\ell \xrightarrow{\eta}_C \mathcal{N}$ owes to $[\![P_1]\!]_\ell \xrightarrow{\eta}_C \mathcal{N}$ or $[\![P_1']\!]_\ell \xrightarrow{\eta}_C \mathcal{N}$. Since $[\![P_1]\!]_\ell \simeq_{rb} [\![P_2]\!]_\ell$ and $[\![P_1']\!]_\ell \simeq_{rb} [\![P_2']\!]_\ell$, there is an $\mathcal{N}'$ such that $[\![P_2]\!]_\ell \xrightarrow{\langle\eta\rangle}_C \mathcal{N}'$ or $[\![P_2']\!]_\ell \xrightarrow{\langle\eta\rangle}_C \mathcal{N}'$ and $\mathcal{N} \simeq_b \mathcal{N}'$. Thus $[\![P_2 + P_2']\!]_\ell \xrightarrow{\eta}_C \mathcal{N}'$ with $\mathcal{N} \simeq_b \mathcal{N}'$.

Case 6. We prove that if $\mathcal{N}_1 \simeq_b \mathcal{N}_2$ then $(\nu\ell)\mathcal{N}_1 \simeq_b (\nu\ell)\mathcal{N}_2$. Let $\mathcal{N}_1 \simeq_b \mathcal{N}_2$ be witnessed by the branching computed network bisimulation relation $\mathcal{R}$. We define $\mathcal{R}' = \{((\nu\ell)\mathcal{N}_1', (\nu\ell)\mathcal{N}_2') | (\mathcal{N}_1', \mathcal{N}_2') \in \mathcal{R}\}$. We prove that $\mathcal{R}'$ is a branching computed network bisimulation relation. Suppose $(\nu\ell)\mathcal{N}_1' \xrightarrow{\eta'}_{C'} (\nu\ell)\mathcal{N}_1''$ resulting from the application of $Rest$ on $\mathcal{N}_1' \xrightarrow{\eta}_C \mathcal{N}_1''$. Since $(\mathcal{N}_1', \mathcal{N}_2') \in \mathcal{R}$, there are two cases; in the first case η is a receive or τ action and $(\mathcal{N}_1'', \mathcal{N}_2') \in \mathcal{R}$, consequently $((\nu\ell)\mathcal{N}_1'', (\nu\ell)\mathcal{N}_2') \in \mathcal{R}'$. In second case there are $\mathcal{N}_2'''$ and $\mathcal{N}_2''$ such that $\mathcal{N}_2' \Rightarrow \mathcal{N}_2''' \xrightarrow{\langle\eta\rangle}_C \mathcal{N}_2''$ with $(\mathcal{N}_1', \mathcal{N}_2'''), (\mathcal{N}_1'', \mathcal{N}_2'') \in \mathcal{R}$. By application of $Rest$, $(\nu\ell)\mathcal{N}_2' \Rightarrow (\nu\ell)\mathcal{N}_2'''$ with $((\nu\ell)\mathcal{N}_1', (\nu\ell)\mathcal{N}_2''') \in \mathcal{R}'$. There are two cases to consider:

- $\langle\eta\rangle = \eta$: Consequently $(\nu\ell)\mathcal{N}_2''' \xrightarrow{\eta'}_{C'} (\nu\ell)\mathcal{N}_2''$.
- $\langle\eta\rangle \neq \eta$: in this case η is of the form $m(\widehat{u})!\{?\}$, $\eta' = \eta$ and $C' = hide(C, \ell)$. If $\langle\eta\rangle = \eta[\ell/?]$ then $\langle\eta\rangle[?/\ell] = \eta$ and $C' = hide(C[\ell/?], \ell)$ hold, otherwise $\langle\eta\rangle[?/\ell] = \langle\eta\rangle$ and $C'[\ell'/?] = hide(C[\ell'/?], \ell)$ hold where $\ell' \neq \ell$. Consequently $(\nu\ell)\mathcal{N}_2''' \xrightarrow{\langle\eta'\rangle}_{C'} (\nu\ell)\mathcal{N}_2''$.

According to the discussion above, there are $\mathcal{N}_2'''$ and $\mathcal{N}_2''$ such that $(\nu\ell)\mathcal{N}_2' \Rightarrow (\nu\ell)\mathcal{N}_2''' \xrightarrow{\langle\eta'\rangle}_{C'} (\nu\ell)\mathcal{N}_2''$ with $((\nu\ell)\mathcal{N}_1', (\nu\ell)\mathcal{N}_2'''), ((\nu\ell)\mathcal{N}_1'', (\nu\ell)\mathcal{N}_2'') \in \mathcal{R}'$.

Likewise we can prove that $\mathcal{N}_1 \simeq_{rb} \mathcal{N}_2$ implies $(\nu\ell)\mathcal{N}_1 \simeq_{rb} (\nu\ell)\mathcal{N}_2$. To this aim we examine the root condition in Definition 6.2. Suppose $(\nu\ell)\mathcal{N}_1 \xrightarrow{\eta'}_{C'} (\nu\ell)\mathcal{N}_1'$. With the same argument as above, $(\nu\ell)\mathcal{N}_2 \xrightarrow{\langle\eta'\rangle}_{C'} (\nu\ell)\mathcal{N}_2'$. Since $\mathcal{N}_1' \simeq_b \mathcal{N}_2'$, we proved that $(\nu\ell)\mathcal{N}_1' \simeq_b (\nu\ell)\mathcal{N}_2'$. Concluding $(\nu\ell)\mathcal{N}_1 \simeq_{rb} (\nu\ell)\mathcal{N}_2$.

Case 9. First we prove that if $\mathcal{N}_1 \simeq_b \mathcal{N}_2$, then $\mathcal{N}_1 \parallel \mathcal{N} \simeq_b \mathcal{N}_2 \parallel \mathcal{N}$. Let $\mathcal{N}_1 \simeq_b \mathcal{N}_2$ be witnessed by the branching computed network bisimulation relation $\mathcal{R}$. We define $\mathcal{R}' = \{(\mathcal{N}_1' \parallel \mathcal{N}', \mathcal{N}_2' \parallel \mathcal{N}') | (\mathcal{N}_1', \mathcal{N}_2') \in \mathcal{R}, \mathcal{N}' \text{ any computed network term}\}$. We prove that $\mathcal{R}'$ is a branching computed network bisimulation relation. Suppose $\mathcal{N}_1 \parallel \mathcal{N} \xrightarrow{\eta}_{C^*} \mathcal{N}^*$. There are several cases to consider:

- Suppose η is a send action $m(\widehat{u})!$ performed by an address ℓ. First let it be performed by $\mathcal{N}_1'$, and $\mathcal{N}$ participated in the communication. That is, $\mathcal{N}_1' \xrightarrow{m(\widehat{u})!\{\ell\}}_{C_1} \mathcal{N}_1''$ and $\mathcal{N} \xrightarrow{m(\widehat{u})?}_{C} \mathcal{N}'$ give rise to the transition $\mathcal{N}_1' \parallel \mathcal{N} \xrightarrow{m(\widehat{u})!\{\ell\}}_{C_1 \cup C[\ell/?]} \mathcal{N}_1'' \parallel \mathcal{N}'$. As $(\mathcal{N}_1', \mathcal{N}_2') \in \mathcal{R}$ and $\mathcal{N}_1' \xrightarrow{m(\widehat{u})!\{\ell\}}_{C_1} \mathcal{N}_1''$, there are $\mathcal{N}_2''$ and $\mathcal{N}_2'''$ such that $\mathcal{N}_2' \Rightarrow \mathcal{N}_2''' \xrightarrow{m(\widehat{u})!\{\ell'\}}_{C_1[\ell'/\ell]} \mathcal{N}_2''$, where $(\ell =? \vee \ell = \ell')$ and $(\mathcal{N}_1', \mathcal{N}_2'''), (\mathcal{N}_1'', \mathcal{N}_2'') \in \mathcal{R}$. Hence $\mathcal{N}_2' \parallel \mathcal{N} \Rightarrow \mathcal{N}_2''' \parallel \mathcal{N} \xrightarrow{m(\widehat{u})!\{\ell'\}}_{C_1 \cup C[\ell'/?]} \mathcal{N}_2'' \parallel \mathcal{N}'$ with $(\mathcal{N}_1' \parallel \mathcal{N}, \mathcal{N}_2''' \parallel \mathcal{N}), (\mathcal{N}_1'' \parallel \mathcal{N}', \mathcal{N}_2'' \parallel \mathcal{N}') \in \mathcal{R}'$.

  Now suppose that the send action was performed by $\mathcal{N}$, and $\mathcal{N}_1'$ participated in the communication. That is, $\mathcal{N}_1' \xrightarrow{m(\widehat{u})?}_{C_1} \mathcal{N}_1''$ and $\mathcal{N} \xrightarrow{m(\widehat{u})!\{\ell\}}_{C} \mathcal{N}'$ give rise to the transition $\mathcal{N}_1' \parallel \mathcal{N} \xrightarrow{m(\widehat{u})!\{\ell\}}_{C_1[\ell/?] \cup C} \mathcal{N}_1'' \parallel \mathcal{N}'$. Since $(\mathcal{N}_1', \mathcal{N}_2') \in \mathcal{R}$ and $\mathcal{N}_1' \xrightarrow{m(\widehat{u})?}_{C_1} \mathcal{N}_1''$, two cases can be considered: either $(\mathcal{N}_1'', \mathcal{N}_2') \in \mathcal{R}$, or there are $\mathcal{N}_2'''$ and $\mathcal{N}_2''$ such that $\mathcal{N}_2' \Rightarrow \mathcal{N}_2''' \xrightarrow{m(\widehat{u})?}_{C_1} \mathcal{N}_2''$ with $(\mathcal{N}_1', \mathcal{N}_2'''), (\mathcal{N}_1'', \mathcal{N}_2'') \in \mathcal{R}$. In the first case by Par and Exe, $\mathcal{N}_2' \parallel \mathcal{N} \xrightarrow{m(\widehat{u})!\{\ell\}}_{C_1 \cup C[\ell/?]} \mathcal{N}_2' \parallel \mathcal{N}'$, and $(\mathcal{N}_1'' \parallel \mathcal{N}', \mathcal{N}_2' \parallel \mathcal{N}') \in \mathcal{R}$. In the second case, $\mathcal{N}_2' \parallel \mathcal{N} \Rightarrow \mathcal{N}_2''' \parallel \mathcal{N} \xrightarrow{m(\widehat{u})!\{\ell\}}_{C_1 \cup C[\ell/?]} \mathcal{N}_2'' \parallel \mathcal{N}'$, and $(\mathcal{N}_1' \parallel \mathcal{N}, \mathcal{N}_2''' \parallel \mathcal{N}), (\mathcal{N}_1'' \parallel \mathcal{N}', \mathcal{N}_2'' \parallel \mathcal{N}') \in \mathcal{R}'$.

  The cases where $\mathcal{N}$ or $\mathcal{N}_1$ does not participate in the communication are straightforward.

- The case where η is a receive action $m(\widehat{u})?$ or a τ action is also straightforward; it originates from $\mathcal{N}_1, \mathcal{N}$, or both in the former case and in the latter case it originates from $\mathcal{N}_1$ or $\mathcal{N}$.

Likewise we can prove that $\mathcal{N}_1 \simeq_{rb} \mathcal{N}_2$ implies $\mathcal{N} \parallel \mathcal{N}_1 \simeq_{rb} \mathcal{N} \parallel \mathcal{N}_2$.

Now let $\mathcal{N}_1 \simeq_{rb} \mathcal{N}_2$. To prove $\mathcal{N}_1|\mathcal{N} \simeq_{rb} \mathcal{N}_2|\mathcal{N}$, we examine the root condition from Definition 6.2. Suppose $\mathcal{N}_1|\mathcal{N} \xrightarrow{m(\widehat{u})!\{\ell\}}_{C^*} \mathcal{N}^*$. There are two cases to consider:

- This send action was performed by $\mathcal{N}_1$ at node ℓ, and $\mathcal{N}$ participated in the communication. That is, $\mathcal{N}_1 \xrightarrow{m(\widehat{u})!\{\ell\}}_{C_1} \mathcal{N}_1'$ and $\mathcal{N} \xrightarrow{m(\widehat{u})?}_{C} \mathcal{N}'$, so that $\mathcal{N}_1|\mathcal{N} \xrightarrow{m(\widehat{u})!\{\ell\}}_{C_1 \cup C[\ell/?]} \mathcal{N}_1' \parallel \mathcal{N}'$. Since $\mathcal{N}_1 \simeq_{rb} \mathcal{N}_2$, there is an $\mathcal{N}_2'$ such that $\mathcal{N}_2 \xrightarrow{m(\widehat{u})!\{\ell'\}}_{C_1[\ell'/\ell]} \mathcal{N}_2'$ with $(\ell =? \vee \ell = \ell')$ and $\mathcal{N}_1' \simeq_b \mathcal{N}_2'$. Then $\mathcal{N}_2|\mathcal{N} \xrightarrow{m(\widehat{u})!\{\ell'\}}_{C_1 \cup C[\ell'/?]} \mathcal{N}_2' \parallel \mathcal{N}'$. Since $\mathcal{N}_1' \simeq_b \mathcal{N}_2'$, we proved that $\mathcal{N}_1' \parallel \mathcal{N}' \simeq_b \mathcal{N}_2' \parallel \mathcal{N}'$.

- The send action was performed $\mathcal{N}$ at node ℓ, and $\mathcal{N}_1$ participated in the communication. That is, $\mathcal{N}_1 \xrightarrow{m(\widehat{u})?}_{C_1} \mathcal{N}_1'$ and $\mathcal{N} \xrightarrow{m(\widehat{u})!\{\ell\}}_{C} \mathcal{N}$, so that $\mathcal{N}_1|\mathcal{N} \xrightarrow{m(\widehat{u})!\{\ell\}}_{C_1 \cup C[\ell/?]} \mathcal{N}_1 \parallel \mathcal{N}'$. Since $\mathcal{N}_1 \simeq_{rb} \mathcal{N}_2$, there is an $\mathcal{N}_2'$ such that $\mathcal{N}_2 \xrightarrow{m(\widehat{u})?}_{C_1} \mathcal{N}_2'$ with $\mathcal{N}_1' \simeq_b \mathcal{N}_2'$. Then $\mathcal{N}_2|\mathcal{N} \xrightarrow{m(\widehat{u})!\{\ell\}}_{C_1 \cup C[\ell/?]} \mathcal{N}_2' \parallel \mathcal{N}'$. Since $\mathcal{N}_1' \simeq_b \mathcal{N}_2'$, we have $\mathcal{N}_1' \parallel \mathcal{N}' \simeq_b \mathcal{N}_2' \parallel \mathcal{N}'$.

Finally, the case where $\mathcal{N}_1|\mathcal{N} \xrightarrow{m(\widehat{u})?}_{C^*} \mathcal{N}^*$ can be easily dealt with. This receive action was performed by both $\mathcal{N}_1$ and $\mathcal{N}$.

Concluding, $\mathcal{N}_1|\mathcal{N} \simeq_{rb} \mathcal{N}_2|\mathcal{N}$. Likewise it can be argued that $\mathcal{N}|\mathcal{N}_1 \simeq_{rb} \mathcal{N}|\mathcal{N}_2$.

Case 10. We prove that if $\mathcal{N}_1 \simeq_b \mathcal{N}_2$, then $\partial_M(\mathcal{N}_1) \simeq_b \partial_M(\mathcal{N}_2)$. Let $\mathcal{N}_1 \simeq_b \mathcal{N}_2$ be witnessed by the branching computed network bisimulation relation $\mathcal{R}$. We define $\mathcal{R}' = \{(\partial_M(\mathcal{N}_1'), \partial_M(\mathcal{N}_2')) | (\mathcal{N}_1', \mathcal{N}_2') \in \mathcal{R}\}$. We prove that $\mathcal{R}'$ is a branching computed network bisimulation relation. Suppose that $\partial_M(\mathcal{N}_1') \xrightarrow{\eta}_C$

$\partial_M(\mathcal{N}_1'')$ results from the application of $Encap$ on $\mathcal{N}_1' \xrightarrow{\eta}_C \mathcal{N}_1''$ such that $Obj(\eta) \notin M$ or η is a send action. Since $(\mathcal{N}_1', \mathcal{N}_2') \in \mathcal{R}$, there are $\mathcal{N}_2'''$ and $\mathcal{N}_2''$ such that $\mathcal{N}_2' \Rightarrow \mathcal{N}_2''' \xrightarrow{\langle\eta\rangle}_C \mathcal{N}_2''$ with $(\mathcal{N}_1', \mathcal{N}_2'''), (\mathcal{N}_1'', \mathcal{N}_1'') \in \mathcal{R}$. Consequently, by application of $Encap$, $\partial_M(\mathcal{N}_2') \Rightarrow \partial_M(\mathcal{N}_2''') \xrightarrow{\langle\eta\rangle}_C \partial_M(\mathcal{N}_2'')$ with $(\partial_M(\mathcal{N}_1'), \partial_M(\mathcal{N}_2''')), (\partial_M(\mathcal{N}_1''), \partial_M(\mathcal{N}_1'')) \in \mathcal{R}'$.

Likewise we can prove that $\mathcal{N}_1 \simeq_{rb} \mathcal{N}_2$ implies $\partial_M(\mathcal{N}_1) \simeq_{rb} \partial_M(\mathcal{N}_2)$. To this aim we examine the root condition in Definition 6.2. Suppose $\partial_M(\mathcal{N}_1) \xrightarrow{\eta}_C \partial_M(\mathcal{N}_1')$. With the same argument as above, $\partial_M(\mathcal{N}_2) \xrightarrow{\langle\eta\rangle}_C \partial_M(\mathcal{N}_2')$. Since $\mathcal{N}_1' \simeq_b \mathcal{N}_2'$, we proved that $\partial_M(\mathcal{N}_1') \simeq_b \partial_M(\mathcal{N}_2')$. Concluding $\partial_M(\mathcal{N}_1) \simeq_{rb} \partial_M(\mathcal{N}_2)$. □

C. Strong versus Rooted Branching Computed Network Bisimilarity

Proposition C.1. Let P_1 and P_2 be two protocol processes, such that $P_1 \simeq P_2$. Then $[\![P_1]\!]_\ell \simeq_{rb} [\![P_2]\!]_\ell$, where ℓ is an arbitrary location.

Proof:
Let $P_1 \simeq P_2$ witnessed by the strong bisimulation relation R. We define $R' = \{([\![P_1']\!]_\ell, [\![P_2']\!]_\ell) | (P_1', P_2') \in R\}$. We prove that R' is a branching computed network bisimulation relation. Suppose $[\![P_1']\!]_\ell \xrightarrow{\eta}_C [\![P_1'']\!]_\ell$ results from the application of $Inter_1$ or $Inter_2$. In the former case, η is of the form $m(\widehat{u})!\{\ell\}$, $C = \{\}$ and $P_1' \xrightarrow{m(\widehat{u})!} P_1''$. Since $(P_1', P_2') \in R$, then there is a P_2'' such that $P_2' \xrightarrow{m(\widehat{u})!} P_2''$ and $(P_1'', P_2'') \in R$. Consequently $[\![P_2']\!]_\ell \xrightarrow{\eta}_C [\![P_2'']\!]_\ell$ and $([\![P_1'']\!]_\ell, [\![P_2'']\!]_\ell) \in R'$. In the latter case, η is of the form $m(\widehat{u})?$ and $C = \{? \rightsquigarrow \ell\}$, and $P_1' \xrightarrow{m(\widehat{u})?} P_1''$. With the same argumentation, $[\![P_2']\!]_\ell \xrightarrow{\eta}_C [\![P_2'']\!]_\ell$ with $([\![P_1'']\!]_\ell, [\![P_2'']\!]_\ell) \in R'$. Thus R' is a branching computed network bisimulation. Hence $P_1 \simeq P_2$ implies $[\![P_1]\!]_\ell \simeq_b [\![P_2]\!]_\ell$, and since $[\![P_1]\!]_\ell$ and $[\![P_2]\!]_\ell$ are matched in each transition step, we can conclude that $[\![P_1]\!]_\ell \simeq_{rb} [\![P_2]\!]_\ell$. □

D. Soundness of the *CNT* Axiomatization

We define branching computed network bisimilation up to $\simeq_b$ (in the same way as [8]). On the one hand it is less restrictive than the notion of a branching computed network bisimulation. On the other hand, if two computed network are related by a branching computed network bisimulation up to $\simeq_b$, they are branching computed network bisimilar (see Proposition D.1). Consequently this notion can be exploited to alleviate soundness proofs of axioms.

Definition D.1. A branching computed network bisimilation up to $\simeq_b$ is a relation $\mathcal{R}$ such that if $\mathcal{N}_1 \mathcal{R} \mathcal{N}_2$ and $\mathcal{N}_1 \Rightarrow \mathcal{N}_1' \xrightarrow{\eta}_C \mathcal{N}_1''$ with $\mathcal{N}_1 \simeq_b \mathcal{N}_1'$ and (η is not a receive or τ action $\vee\ \mathcal{N}_1' \not\simeq_b \mathcal{N}_1''$) then there exist $\mathcal{N}_1^{*'}, \mathcal{N}_1^{*''}, \mathcal{N}_2^{*'}, \mathcal{N}_2^{*''}, \mathcal{N}_2', \mathcal{N}_2''$ such that $\mathcal{N}_2 \Rightarrow \mathcal{N}_2' \xrightarrow{\langle\eta\rangle}_C \mathcal{N}_2''$ with

$$\mathcal{N}_1' \simeq_b \mathcal{N}_1^{*'} \wedge \mathcal{N}_2' \simeq_b \mathcal{N}_2^{*'} \wedge \mathcal{N}_1^{*'} \mathcal{R} \mathcal{N}_2^{*'}$$
$$\mathcal{N}_1'' \simeq_b \mathcal{N}_1^{*''} \wedge \mathcal{N}_2'' \simeq_b \mathcal{N}_2^{*''} \wedge \mathcal{N}_1^{*''} \mathcal{R} \mathcal{N}_2^{*''}.$$

Similarly the converse must hold if $\mathcal{N}_2 \Rightarrow \mathcal{N}_2' \xrightarrow{\eta}_C \mathcal{N}_2''$.

Proposition D.1. If $\mathcal{R}$ is a branching computed network bisimulation up to $\simeq_b$ and $\mathcal{N}_1 \mathcal{R} \mathcal{N}_2$, then $\mathcal{N}_1 \simeq_b \mathcal{N}_2$.

Proof:
It suffices to prove that the relation $\simeq_b \mathcal{R} \simeq_b = \{(\mathcal{N}_1^*, \mathcal{N}_2^*) | \exists \mathcal{N}_1, \mathcal{N}_2 : \mathcal{N}_1^* \simeq_b \mathcal{N}_1 \mathcal{R} \mathcal{N}_2 \simeq_b \mathcal{N}_2^*\}$ is a branching computed network bisimulation. So suppose $\mathcal{N}_1^*, \mathcal{N}_1, \mathcal{N}_2$ and $\mathcal{N}_2^*$ are as indicated, and $\mathcal{N}_1^* \xrightarrow{\eta}_C \mathcal{N}_1^{*''}$. Then either η is a receive or τ action and $\mathcal{N}_1^{*''} \simeq_b \mathcal{N}_1$ which completes the proof, or there are $\mathcal{N}_1'$ and $\mathcal{N}_1''$ with $\mathcal{N}_1' \simeq_b \mathcal{N}_1^* \simeq_b \mathcal{N}_1$ and $\mathcal{N}_1'' \simeq_b \mathcal{N}_1^{*''} (\not\simeq_b \mathcal{N}_1'$ if η is a receive or τ action). By Definition D.1, there are $\mathcal{N}_2', \mathcal{N}_2''$ such that $\mathcal{N}_2 \Rightarrow \mathcal{N}_2' \xrightarrow{\langle\eta\rangle}_C \mathcal{N}_2''$ with $(\mathcal{N}_1', \mathcal{N}_2'), (\mathcal{N}_1'', \mathcal{N}_2'') \in \simeq_b \mathcal{R} \simeq_b$. Since $\mathcal{N}_2^* \simeq_b \mathcal{N}_2$, by application of Lemma A.1 and Definition 6.1, there are $\mathcal{N}_2^{*'}$ and $\mathcal{N}_2^{*''}$ such that $\mathcal{N}_2^* \Rightarrow \mathcal{N}_2^{*'} \xrightarrow{\langle\eta\rangle} \mathcal{N}_2^{*''}$ with $\mathcal{N}_2' \simeq_b \mathcal{N}_2^{*'}$ and $\mathcal{N}_2'' \simeq_b \mathcal{N}_2^{*''}$. Consequently, $\mathcal{N}_2 \Rightarrow \mathcal{N}_2' \xrightarrow{\langle\eta\rangle}_C \mathcal{N}_2''$ with $(\mathcal{N}_1^*, \mathcal{N}_2^{*'}), (\mathcal{N}_1^{*''}, \mathcal{N}_2^{*''}) \in \simeq_b \mathcal{R} \simeq_b$. The same argumentation holds when $\mathcal{N}_2^* \xrightarrow{\eta}_C \mathcal{N}_2^{*''}$ □

To prove Theorem 7.1, it suffices to prove soundness of each axiom modulo rooted branching computed network bisimilarity separately. For axiom Con, every initial transition of $C_1\eta.\mathcal{N}$ is obviously also an initial transition of $C_1\eta.\mathcal{N} + C_2\eta.\mathcal{N}$. Vice versa, if $C_1\eta.\mathcal{N} + C_2\eta.\mathcal{N}$ can perform an initial transition because $C_2\eta.\mathcal{N}$ can, then by application of the Pre and Exe rules in Table 2, $C_1\eta.\mathcal{N}$ can perform this initial transition too. Similarly, it is not hard to argue the soundness of $P_{1-5}, R, Dead, Obs, Choice_{1-4}$, Br, LEx_{1-3}, S_{1-4}, $Sync_{1-5}$, Res_{1-4}, Abs_{1-3}, and Ecp_{1-4}, by showing that the terms on both sides of each axiom satisfy the transfer conditions of Definition 6.2. Soundness of axiom P_0 is the direct result of Proposition 6.1.

We focus on the soundness of T_1 and T_2. We only explain the soundness T_2, as the soundness of T_1 can be argued in a similar fashion. The only transition the terms $C\eta.(C'\tau.(\mathcal{N}_1 + \mathcal{N}_2) + \mathcal{N}_2)$ and $C\eta.(\mathcal{N}_1 + \mathcal{N}_2)$ in T_2 can do is $\xrightarrow{\eta}_C$, and the resulting terms $C'\tau.(\mathcal{N}_1 + \mathcal{N}_2) + \mathcal{N}_2$ and $\mathcal{N}_1 + \mathcal{N}_2$ are branching computed network bisimilar, witnessed by the relation $\mathcal{R}$ constructed as follows:

$$\mathcal{R} = \{(C'\tau.(\mathcal{N}_1 + \mathcal{N}_2) + \mathcal{N}_2, \mathcal{N}_1 + \mathcal{N}_2), (\mathcal{N}, \mathcal{N}) | \mathcal{N} \in CNT\}.$$

The pair $(C'\tau.(\mathcal{N}_1 + \mathcal{N}_2) + \mathcal{N}_2, \mathcal{N}_1 + \mathcal{N}_2)$ satisfies the transfer conditions in Definition 6.1. Because every initial transition that $C'\tau.(\mathcal{N}_1 + \mathcal{N}_2) + \mathcal{N}_2$ can perform owing to $\mathcal{N}_2$, $\mathcal{N}_1 + \mathcal{N}_2$ can perform too. And after the initial $\xrightarrow{\tau}_{C'}$-transition, $(\mathcal{N}_1 + \mathcal{N}_2, \mathcal{N}_1 + \mathcal{N}_2) \in \mathcal{R}$ holds. And every initial transition $\mathcal{N}_1 + \mathcal{N}_2$ can perform, $C'\tau.(\mathcal{N}_1 + \mathcal{N}_2) + \mathcal{N}_2$ can mimic, by first doing its $\xrightarrow{\tau}_{C'}$-transition.

Soundness of the axiom $Unfold$ follows directly from the rule Rec, since $rec\mathcal{A}_n.t \xrightarrow{\eta}_C \mathcal{N} \Leftrightarrow t\{rec\mathcal{A}_n.t/t\} \xrightarrow{\eta}_C \mathcal{N}$. Soundness of axiom $Fold$ is a consequence of the following proposition, taken from [8]:

Proposition D.2. If $s \simeq_{rb} t\{s/\mathcal{A}_n\}$ then $s \simeq_{rb} rec\mathcal{A}_n.t$, provided $\mathcal{A}_n$ is guarded in t.

Proof:
For $u, v \in CNT$ we write $u(v)$ for $u\{v/\mathcal{A}_n\}$. Assume $s, t, v \in CNT$ such that $\mathcal{A}_n$ is guarded in t, $s \simeq_{rb} t(s)$ and $v \simeq_{rb} t(v)$. We will prove that $\{(u(t(s))), u(t(v)))) | u \in CNT\}$ is a branching computed network bisimulation up to $\simeq_b$. Since $\mathcal{A}_n$ is guarded in t and hence in $u(t)$, the first transition with subscript C and label η where $\eta \neq \tau$ generated by $u(t(s))$ does not originate from s (which can be easily shown by induction on the inference tree), so $u(t(s)) \Rightarrow u'(s) \xrightarrow{\eta}_C u''(s)$. Then also

$u(t(v)) \Rightarrow u'''(v) \xrightarrow{\eta}_C u''''(v)$ with $u''' \equiv_\alpha u'$ and $u'''' \equiv_\alpha u''$. It should be noted that since the free network names of s and v may be different, their substitution in $u(t)$ may result in different renamings of bound names in $u(t)$. Due to congruence of rooted branching computed network bisimilarity and that $\simeq_{rb}$ implies $\simeq_b$, we get $u'(t(s)) \simeq_b u'(s)$, $u''(t(s)) \simeq_b u''(s)$, $u'''(t(v)) \simeq_b u'''(v)$, and $u''''(t(v)) \simeq_b u''''(v)$. The requirement starting with $u(t(v))$ follows by symmetry, so the relation is a branching computed network bisimulation up to $\simeq_b$, and by proposition D.1, $u(t(s)) \simeq_b u(t(v))$. Work it out considering $u'(t(s)) = u(t(s))$, we can prove that $u(t(s)) \simeq_{rb} u(t(v))$, so since u is an arbitrary term, we have in particular $t(s) \simeq_{rb} t(v)$ and hence $s \simeq_{rb} t(s) \simeq_{rb} t(v) \simeq_{rb} v$. Finally, take $v = rec\mathcal{A}_n.t$; note that $rec\mathcal{A}_n.t \simeq_{rb} t(rec\mathcal{A}_n.t)$. □

Soundness of axiom Ung follows by application of Rec: $rec\mathcal{A}_n.(\mathcal{A}_n + t) \xrightarrow{\eta}_C t'\{L/\mathcal{A}_n\} \Leftrightarrow rec\mathcal{A}_n.t \xrightarrow{\eta}_C t'\{R/\mathcal{A}_n\}$, and proving that $t'\{L/\mathcal{A}_n\} \simeq_b t'\{R/\mathcal{A}_n\}$, where L and R are the left- and right-hand sides of Ung_2. It is straightforward to show that $\mathcal{R} = \{(t\{L/\mathcal{A}_n\}, t\{R/\mathcal{A}_n\}) | t \in CNT_f\}$ is a branching computed network bisimulation relation. In the same approach, soundness of axioms $WUng_1$ and $WUng_2$ holds since the following relations are branching computed network bisimulations:

$$\begin{aligned}\mathcal{R}' = {} & \{((C'\tau.t' + t)\{L/\mathcal{A}_n\}, (t' + t)\{R/\mathcal{A}_n\}), (t'\{L/\mathcal{A}_n\}, (t + t')\{R/\mathcal{A}_n\})\} \cup \\ & \{(t\{L/\mathcal{A}_n\}, t\{R/\mathcal{A}_n\}) | t \in CNT_f\} \\ \mathcal{R}'' = {} & \{((\mathcal{A}_n + t)\{L/\mathcal{A}_n\}, (t + s)\{L/\mathcal{A}_n\})\} \cup \{(t\{L/\mathcal{A}_n\}, t\{R/\mathcal{A}_n\}) | t \in CNT_f\}\end{aligned}$$

where L and R are the left- and right-hand sides of corresponding axioms. To prove $\mathcal{R}'$ is a branching computed network bisimulation, it suffices to show that $(t'\{L/\mathcal{A}_n\}, (t + t')\{R/\mathcal{A}_n\})$ satisfies the transfer conditions in Definition 6.1. We have $(t' + t)\{R/\mathcal{A}_n\} \xrightarrow{\eta}_C t''\{R/\mathcal{A}_n\}$, owing to $t\{R/\mathcal{A}_n\} \xrightarrow{\eta}_C t''\{R/\mathcal{A}_n\}$ by application of $Choice'$. Since $\mathcal{A}_n$ is unguarded in t', it is easy to show that

$$t'\{L/\mathcal{A}_n\} \Rightarrow \mathcal{A}_n\{L/\mathcal{A}_n\} \xrightarrow{\tau}_C (C'\tau.t' + t)\{L/\mathcal{A}_n\}$$

and consequently $t'\{L/\mathcal{A}_n\} \Rightarrow (C'\tau.t' + t)\{L/\mathcal{A}_n\} \xrightarrow{\eta}_C t''\{L/\mathcal{A}_n\}$, with $((C'\tau.t' + t)\{L/\mathcal{A}_n\}, (t' + t)\{R/\mathcal{A}_n\}) \in \mathcal{R}'$ and $(t''\{L/\mathcal{A}_n\}, t''\{R/\mathcal{A}_n\}) \in \mathcal{R}'$. Conversely every transition performed by $t'\{L/\mathcal{A}_n\}$ can be performed by $(t + t')\{R/\mathcal{A}_n\}$, while their resulted terms are included in $\mathcal{R}'$. $\mathcal{R}''$ is proved in the same way by application of Rec.

To prove soundness of axiom Hid, we show that the following relation is a branching computed network bisimulation up to $\simeq_b$:

$$\begin{aligned}\mathcal{R}''' = \{(\nabla_M(s\{rec\mathcal{A}_n.t/\mathcal{A}_n\}), \nabla_M(s\{rec\mathcal{A}_n.\nabla_M(t)/\mathcal{A}_n\})) \\ | s, t \in CNT \text{ and only } \mathcal{A}_n \text{ is free in } s\}\end{aligned}$$

To this aim we show that $\mathcal{R}'''$ satisfies the following transfer condition: if $\nabla_M(s\{rec\mathcal{A}_n.t/\mathcal{A}_n\}) \xrightarrow{\eta}_C u$ then, for some u' and u'', $\nabla_M(s\{rec\mathcal{A}_n.\nabla_M(t)/\mathcal{A}_n\}) \xrightarrow{\eta}_C u''$, $u'' \simeq_b u'$ and $(u, u') \in \mathcal{R}'''$, and symmetrically for transitions of $\nabla_M(s\{rec\mathcal{A}_n.\nabla_M(t)/\mathcal{A}_n\})$.

Being a branching computed network bisimulation up to $\simeq_b$ for $\mathcal{R}'''$ implies

$$\nabla_M(t\{rec\mathcal{A}_n.t/\mathcal{A}_n\}) \simeq_b \nabla_M(t\{\mathcal{A}_n.\nabla_M(t)/\mathcal{A}_n\})$$

which implies $\nabla_M(rec\mathcal{A}_n.t) \simeq_{rb} rec\mathcal{A}_n.\nabla_M(t)$, because $\nabla_M(rec\mathcal{A}_n.t) \xrightarrow{\eta}_C u$ if and only if $\nabla_M(t\{rec\mathcal{A}_n.t/\mathcal{A}_n\}) \xrightarrow{\eta}_C u$, and similarly, $rec\mathcal{A}_n.\nabla_M(t) \xrightarrow{\eta}_C u$ if and only if $\nabla_M(t\{\mathcal{A}_n.\nabla_M(t)/\mathcal{A}_n\}) \xrightarrow{\eta}_C u$.

We prove that $\mathcal{R}'''$ satisfies the above transfer condition, by induction on the height of the inference tree by which $C\eta$ transitions of $\nabla_M(s\{rec\mathcal{A}_n.t/\mathcal{A}_n\})$ are inferred.

The base cases of the induction are the following ones:

- if $s \equiv 0$ or $s \equiv [\![0]\!]_\ell$, then the condition above trivially holds.
- if $s \equiv [\![P]\!]_{\ell'}$, then $\nabla_M(s\{rec\mathcal{A}_n.t/\mathcal{A}_n\}) \equiv \nabla_M([\![P]\!]_{\ell'}) \equiv \nabla_M(s\{rec\mathcal{A}_n.\nabla_M(t)/\mathcal{A}_n\})$ which trivially holds.
- if $s \equiv C\eta.s'$, then $\nabla_M(C\eta.s'\{rec\mathcal{A}_n.t/\mathcal{A}_n\}) \xrightarrow{\delta_M(\eta)}_C \nabla_M(s'\{rec\mathcal{A}_n.t/\mathcal{A}_n\})$.
  Also $\nabla_M(C\eta.s'\{rec\mathcal{A}_n.\nabla_M(t)/\mathcal{A}_n\}) \xrightarrow{\delta_M(\eta)}_C \nabla_M(s'\{rec\mathcal{A}_n.\nabla_M(t)/\mathcal{A}_n\})$, and the targets are related by $\mathcal{R}'''$.

We now consider the inductive steps. We have the following cases, based on the structure of s:

- if $s \equiv \mathcal{A}_n$, then $\nabla_M(s\{rec\mathcal{A}_n.t/\mathcal{A}_n\}) \equiv \nabla_M(rec\mathcal{A}_n.t)$ and $\nabla_M(s\{rec\mathcal{A}_n.\nabla_M(t)/\mathcal{A}_n\}) \equiv \nabla_M(rec\mathcal{A}_n.\nabla_M(t))$. Since $\nabla_M(rec\mathcal{A}_n.t) \xrightarrow{\delta_M(\eta)}_C u$, it must be that $rec\mathcal{A}_n.t \xrightarrow{\eta}_C v$ with $u \equiv \nabla_M(v)$. Furthermore, it must be that $t\{rec\mathcal{A}_n.t/\mathcal{A}_n\} \xrightarrow{\eta}_C v$ by a shorter inference. As a consequence we derive $\nabla_M(t\{rec\mathcal{A}_n.t/\mathcal{A}_n\}) \xrightarrow{\delta_M(\eta)}_C u$.
  By induction we derive $\nabla_M(t\{rec\mathcal{A}_n.\nabla_M(t)/\mathcal{A}_n\}) \xrightarrow{\delta_M(\eta)}_C u''$ with $u'' \simeq_b u'$ and $(u, u') \in \mathcal{R}'''$. As a consequence $rec\mathcal{A}_n.\nabla_M(t) \xrightarrow{\delta_M(\eta)}_C u''$ and $\nabla_M(rec\mathcal{A}_n.\nabla_M(t)) \xrightarrow{\delta_M(\eta)}_C \nabla_M(u'')$. Since u'' has abstraction as the outermost operator (because it is derived by a transition from a term that has abstraction as the outermost operator), we also have that $\nabla_M(u'') = u''$.
- if $s \equiv s' + s''$, then $\nabla_M(s\{rec\mathcal{A}_n.t/\mathcal{A}_n\}) \equiv \nabla_M(s'\{rec\mathcal{A}_n.t/\mathcal{A}_n\} + s''\{rec\mathcal{A}_n.t/\mathcal{A}_n\})$ and $\nabla_M(s\{rec\mathcal{A}_n.\nabla_M(t)/\mathcal{A}_n\}) \equiv \nabla_M(s'\{rec\mathcal{A}_n.\nabla_M(t)/\mathcal{A}_n\} + s''\{rec\mathcal{A}_n.\nabla_M(t)/\mathcal{A}_n\})$.
  Since $\nabla_M(s'\{rec\mathcal{A}_n.t/\mathcal{A}_n\} + s''\{rec\mathcal{A}_n.t/\mathcal{A}_n\}) \xrightarrow{\delta_M(\eta)}_C u$, it must be that $s'\{rec\mathcal{A}_n.t/\mathcal{A}_n\} + s''\{rec\mathcal{A}_n.t/\mathcal{A}_n\} \xrightarrow{\eta}_C v$ with $u \equiv \nabla_M(v)$. Now we have two cases:
  - if $s'\{rec\mathcal{A}_n.t/\mathcal{A}_n\} \xrightarrow{\eta}_C v$, then $\nabla_M(s'\{rec\mathcal{A}_n.t/\mathcal{A}_n\}) \xrightarrow{\delta_M(\eta)}_C u$, and by induction $\nabla_M(s'\{rec\mathcal{A}_n.\nabla_M(t)/\mathcal{A}_n\}) \xrightarrow{\delta_M(\eta)}_C u''$ with $u'' \simeq_b u'$ and $(u, u') \in \mathcal{R}'''$.
    Therefore it must be that $s'\{rec\mathcal{A}_n.\nabla_M(t)/\mathcal{A}_n\} \xrightarrow{\eta}_C v''$ with $u'' \equiv \nabla_M(v'')$. As a consequence $\nabla_M(s\{rec\mathcal{A}_n.\nabla_M(t)/\mathcal{A}_n\}) \xrightarrow{\eta}_C u''$.
  - if $s''\{rec\mathcal{A}_n.t/\mathcal{A}_n\} \xrightarrow{\eta}_C v$, then the result is derived in a similar way.
- if $s \equiv rec\mathcal{A}_m.t'$, with $\mathcal{A}_m \neq \mathcal{A}_n$, then $\nabla_M(s\{rec\mathcal{A}_n.t/\mathcal{A}_n\}) \equiv \nabla_M(rec\mathcal{A}_m.t'\{rec\mathcal{A}_n.t/\mathcal{A}_n\})$ and $\nabla_M(s\{rec\mathcal{A}_n.\nabla_M(t)/\mathcal{A}_n\}) \equiv \nabla_M(rec\mathcal{A}_m.t'\{rec\mathcal{A}_n.\nabla_M(t)/\mathcal{A}_n\})$. For the case $\mathcal{A}_m = \mathcal{A}_n$, $\nabla_M(s\{rec\mathcal{A}_n.t/\mathcal{A}_n\}) \equiv \nabla_M(s\{rec\mathcal{A}_n.\nabla_M(t)/\mathcal{A}_n\} \equiv \nabla_M(s)$, which was proved previously.

Since $\nabla_M(s\{rec\mathcal{A}_n.t/\mathcal{A}_n\}) \xrightarrow{\delta_M(\eta)}_C u$, it must be that $rec\mathcal{A}_m.t'\{rec\mathcal{A}_n.t/\mathcal{A}_n\}) \xrightarrow{\eta}_C v$ with $u \equiv \nabla_M(v)$. Hence $t'\{rec\mathcal{A}_m.t'/\mathcal{A}_m\}\{rec\mathcal{A}_n.t/\mathcal{A}_n\} \xrightarrow{\eta}_C v$. As a consequence

$$\nabla_M(t'\{rec\mathcal{A}_m.t'/\mathcal{A}_m\}\{rec\mathcal{A}_n.t/\mathcal{A}_n\}) \xrightarrow{\delta_M(\eta)}_C u.$$

By induction $\nabla_M(t'\{rec\mathcal{A}_m.t'/\mathcal{A}_m\}\{rec\mathcal{A}_n.\nabla_M(t)/\mathcal{A}_n\}) \xrightarrow{\delta_M(\eta)}_C u''$ with $u'' \simeq_b u'$ and $(u, u') \in \mathcal{R}'''$. Therefore $t'\{rec\mathcal{A}_m.t'/\mathcal{A}_m\}\{rec\mathcal{A}_n.\nabla_M(t)/\mathcal{A}_n\} \xrightarrow{\eta}_C v''$ with $u'' \equiv \nabla_M(v'')$. As a consequence $rec\mathcal{A}_m.t'\{rec\mathcal{A}_n.\nabla_M(t)/\mathcal{A}_n\} \xrightarrow{\eta}_C v''$, and finally $\nabla_M(s\{rec\mathcal{A}_n.\nabla_M(t)/\mathcal{A}_n\}) \xrightarrow{\eta}_C u''$.

- If $s \equiv s' \parallel s''$ or $s \equiv s' \between s''$ or $s \equiv s' \mid s''$ or $s \equiv \partial_M(s')$ or $s \equiv \nabla_M(s)$ then the condition trivially holds because $\mathcal{A}_n$ cannot occur inside s' or s''.

A symmetric inductive proof is performed when we start from $C\eta$ transitions of $\nabla_M(s\{rec\mathcal{A}_n.\nabla_M(t)/\mathcal{A}_n\})$ in the conditions above.

E. CNT_f Generates Finite-State Behaviors

Proposition E.1. Let $\mathcal{N}$ be a closed CNT_f term such that every subterm $rec\mathcal{A}_n.t$ is essentially finite-state. Given a data model $\mathbb{D}$ with finite data domains, the transition system for $\mathcal{N}$ generated by the operational rules has only finitely many states.

Proof:
We extend the proof strategy used in [8]. First we introduce *Colored Computed Network Theory* (CNT_f^c), where each occurrence of an operator (except static operators, i.e. deployment, parallel, left merge, communication merge, restriction, abstraction, and encapsulation operators) and of a name in a process term has a color (black or red). We color a term using functions $red(\mathcal{N})$ and $black(\mathcal{N})$, which color all occurrences of operators (excluding static ones) and names in $\mathcal{N}$ red and black respectively. For instance, $red(C\eta.0+\mathcal{N}) \equiv red(C\eta.)red(0)red(+)red(\mathcal{N})$. Furthermore, if in a subterm $\alpha.P$, P_1+P_2, $[u_1 = u_2]P_1, P_2$, $C\eta.\mathcal{N}$, $\mathcal{N}_1+\mathcal{N}_2$ or $rec\mathcal{A}_n.t$ the leading operator is colored black, the entire term must be black. Renaming of bound network names does not change their color. Second we introduce an auxiliary transition relation $\rightarrow$ on CNT_f^c such that each $\xrightarrow{\eta}_C$ generated by the CNT_f operational semantics can be matched by a sequence of $\rightarrow$ relations. Therefore it suffices to prove that the set of closed terms reachable by a colored version of any closed $\mathcal{N} \in CNT_f$ under the transition relation $\rightarrow$ is finite. The intuition behind coloring terms is to distinguish between two types of terms reachable under the transitions relation $\rightarrow$; we color a term black to express that its red version has been visited before.

Consider the transition relation $\rightarrow \subseteq CNT_f \times \{., (\widehat{u}), +, [\,], rec\} \times CNT_f$, where $\widehat{u}$ is a sequence of

data terms in $\mathbb{D}$, defined by:

$$\begin{array}{l}
[\![m(\widehat{u})!.P]\!]_\ell \stackrel{\cdot}{\rightarrow} [\![P]\!]_\ell;\\
[\![m(\widehat{x})?.P]\!]_\ell \stackrel{(\widehat{u})}{\longrightarrow} [\![P\{\widehat{u}/\widehat{x}\}]\!]_\ell, \text{ where } \widehat{u} \in domain_m;\\
[\![P_1 + P_2]\!]_\ell \stackrel{+}{\rightarrow} [\![P_1]\!]_\ell \text{ and } [\![P_1 + P_2]\!]_\ell \stackrel{+}{\rightarrow} [\![P_2]\!]_\ell\\
[\![[u]P_1, P_2]\!]_\ell \stackrel{[\,]}{\rightarrow} [\![P_1]\!]_\ell, \text{ if } \mathbb{D} \vdash u = T;\\
[\![[u]P_1, P_2]\!]_\ell \stackrel{[\,]}{\rightarrow} [\![P_2]\!]_\ell, \text{ if} \mathbb{D} \vdash u = F;\\
[\![\mathcal{A}_p(\widehat{u})]\!]_\ell \stackrel{rec}{\longrightarrow} [\![P\{\widehat{u}/\widehat{x}\}]\!]_\ell, \text{ where } \mathcal{A}_p(\langle x : D\rangle) = P;\\
C\eta.\mathcal{N} \stackrel{\cdot}{\rightarrow} \mathcal{N};\\
\mathcal{N}_1 + \mathcal{N}_2 \stackrel{+}{\rightarrow} \mathcal{N}_1 \text{ and } \mathcal{N}_1 + \mathcal{N}_2 \stackrel{+}{\rightarrow} \mathcal{N}_2;\\
\mathcal{N}_1 \parallel \mathcal{N}_2 \stackrel{\delta}{\rightarrow} \mathcal{N}_1' \parallel \mathcal{N}_2 \text{ and } \mathcal{N}_1 \parallel \mathcal{N}_2 \stackrel{\delta}{\rightarrow} \mathcal{N}_1 \parallel \mathcal{N}_2' \text{ if } \mathcal{N}_1 \stackrel{\delta}{\rightarrow} \mathcal{N}_1' \text{ and } \mathcal{N}_2 \stackrel{\delta}{\rightarrow} \mathcal{N}_2' \text{ respectively};\\
\mathcal{N}_1 \mid \mathcal{N}_2 \stackrel{\delta}{\rightarrow} \mathcal{N}_1' \parallel \mathcal{N}_2 \text{ and } \mathcal{N}_1 \mid \mathcal{N}_2 \stackrel{\delta}{\rightarrow} \mathcal{N}_1 \parallel \mathcal{N}_2' \text{ if } \mathcal{N}_1 \stackrel{\delta}{\rightarrow} \mathcal{N}_1' \text{ and } \mathcal{N}_2 \stackrel{\delta}{\rightarrow} \mathcal{N}_2' \text{ respectively};\\
\mathcal{N}_1 \between \mathcal{N}_2 \stackrel{\delta}{\rightarrow} \mathcal{N}_1' \parallel \mathcal{N}_2 \text{ if } \mathcal{N}_1 \stackrel{\delta}{\rightarrow} \mathcal{N}_1';\\
(\nu\ell)\mathcal{N} \stackrel{\delta}{\rightarrow} (\nu\ell)\mathcal{N}', \text{ and } \nabla_M(\mathcal{N}) \stackrel{\delta}{\rightarrow} \nabla_M(\mathcal{N}'), \text{ and } \partial_M(\mathcal{N}) \stackrel{\delta}{\rightarrow} \partial_M(\mathcal{N}'), \text{ if } \mathcal{N} \stackrel{\delta}{\rightarrow} \mathcal{N}';\\
rec\mathcal{A}_n.t \stackrel{rec}{\longrightarrow} t\{rec\mathcal{A}_n.t/\mathcal{A}_n\};
\end{array}$$

where $\delta \in \{., (\widehat{u}), +, [\,], rec\}$. We use $\mathcal{N} \rightarrow^* \mathcal{N}'$ to denote that $\mathcal{N}'$ is reachable from $\mathcal{N}$ under the $\rightarrow$ relation.

The above relation can be defined for colored terms with minor changes: each rule should be defined for any coloring of each operator, while colors of terms are preserved under all transitions except the sixth and thirteenth where the black version of a protocol name and recursive term is substituted for a protocol and (free) network name:

$$\begin{array}{l}
[\![black(\mathcal{A}_p(\widehat{u}))]\!]_\ell \stackrel{rec}{\longrightarrow} [\![black(P\{\widehat{u}/\widehat{x}\})]\!]_\ell;\\
[\![red(\mathcal{A}_p(\widehat{u}))]\!]_\ell \stackrel{rec}{\longrightarrow} [\![red(P\{\widehat{u}/\widehat{x}\})\{black(\mathcal{A}_p(\widehat{u}))/red(\mathcal{A}_p(\widehat{u}))\}]\!]_\ell;\\
black(rec\mathcal{A}_n).t \stackrel{rec}{\longrightarrow} t\{black(rec\mathcal{A}_n.t)/black(\mathcal{A}_n)\};\\
red(rec\mathcal{A}_n).t \stackrel{rec}{\longrightarrow} t\{black(rec\mathcal{A}_n.t)/black(\mathcal{A}_n)\}\{black(rec\mathcal{A}_n.t)/red(\mathcal{A}_n)\}.
\end{array}$$

If $\mathcal{N} \stackrel{\eta}{\rightarrow}_C \mathcal{N}'$ for $\mathcal{N}, \mathcal{N}' \in CNT_f$, and $\mathcal{N}_0$ is a colored version of $\mathcal{N}$, then there must be $\mathcal{N}_1, \cdots, \mathcal{N}_{n+1} \in CNT_f^c$ with $n \in \mathbb{N}$, such that $\mathcal{N}_{i-1} \stackrel{\Delta}{\rightarrow} \mathcal{N}_i$ where $\Delta \in \{., (\widehat{u}), +, rec, [\,]\}$ for all $i = 1, \cdots, n$, and $\mathcal{N}_n \stackrel{\cdot}{\rightarrow} \mathcal{N}_{n+1}$ or $\mathcal{N}_n \stackrel{(\widehat{u})}{\longrightarrow} \mathcal{N}_{n+1}$, and $\mathcal{N}_{n+1}$ is the colored version of $\mathcal{N}'$. Since the data domains are finite in $\mathbb{D}$, there are finitely many transitions labeled by data terms $\widehat{u}$.

Fix an $\mathcal{M} \in CNT_f$. Let $\varepsilon_\mathcal{M}$ denote the set of closed terms that are reachable from $\mathcal{M}$. Likewise, let $\varepsilon'_\mathcal{M}$ denote the set of colored terms that are reachable by $\rightarrow$ from $red(\mathcal{M})$. For any $\mathcal{N} \in \varepsilon_\mathcal{M}$, a colored version appears in $\varepsilon'_\mathcal{M}$, and it suffices to prove that $\varepsilon'_\mathcal{M}$ is finite.

It should be noted that if a term $\mathcal{N}$ is partly red and $\mathcal{N} \rightarrow^* \mathcal{N}'$, then the red part of $\mathcal{N}'$ is smaller than or equal to the red part of $\mathcal{N}$. Also for a red (sub)term $\mathcal{N}$ and $\mathcal{N} \rightarrow \mathcal{N}'$, the red part of $\mathcal{N}'$ is smaller than the red part of $\mathcal{N}$. Thus for a partly red term $\mathcal{N}$ and $\mathcal{N} \rightarrow \mathcal{N}'$, if the red part of $\mathcal{N}'$ is smaller than $\mathcal{N}$, then this transition owes to a transition of a red subterm in $\mathcal{N}$, while if the red part of $\mathcal{N}'$ is equal to $\mathcal{N}$, then this transition owes to a transition of a black subterm in $\mathcal{N}$.

Furthermore for any term $\mathcal{N} \in \varepsilon'_{\mathcal{M}}$, if $\mathcal{N}$ contains a subterm $rec\mathcal{A}_n.t$ with $rec\mathcal{A}_n$ red, then no black subterm of t contains a free occurrence $\mathcal{A}_n$. Since $\mathcal{N}$ is essentially finite-state, no black subterm t contains parallel, communication merge, left merge, restriction, abstraction, and encapsulation operators. These properties are trivially true for $red(\mathcal{M})$, trivially preserved under $\xrightarrow{\Delta}$ where $\Delta = \{., (\widehat{u}), +, [\,]\}$, and preserved under $\xrightarrow{rec}$ by renaming of bound network names. It follows that each $\mathcal{N} \in \varepsilon'_{\mathcal{M}}$ is of the form $\triangle(\triangle_1(\mathcal{N}_1) \odot \cdots \odot \triangle_i(\mathcal{N}_i) \odot \cdots \odot \triangle_n(\mathcal{N}_n))$ where $\odot \in \{\|, |, \between\}$, $\triangle ::= ((\nu\ell) + \nabla_M + \partial_M)^*$ is a sequence of restriction, abstraction and encapsulation operators for some arbitrary $\ell \in Loc$, $M \subseteq Msg$, and for all $1 \leq i \leq n$, each $\mathcal{N}_i$ contains no $\odot$ operators (since recursion subterms are essentially finite-state). Moreover if $\mathcal{N} \rightarrow^* \mathcal{N}'$, then the black subterms of $\mathcal{N}$ that are inherited by $\mathcal{N}'$–unlike the red ones–are unchanged in $\mathcal{N}'$. Thus if a term $\mathcal{N} \in \varepsilon'_{\mathcal{M}}$ which is of the form $\triangle(\triangle_1(\mathcal{N}_1) \odot \cdots \odot \triangle_i(\mathcal{N}_i) \odot \cdots \odot \triangle_n(\mathcal{N}_n))$ where $\mathcal{N}_i$ is partly red, $\mathcal{N} \rightarrow^* \mathcal{N}'$, and $\mathcal{N}'$ is of the form $\triangle(\triangle_1(\mathcal{N}'_1) \parallel \cdots \parallel \triangle_i(\mathcal{N}'_i) \parallel \cdots \parallel \triangle_n(\mathcal{N}'_n))$ and $\mathcal{N}'_i$ is completely black, then $\mathcal{N}'_i$ has the form of $[\![\mathcal{A}_p(\widehat{u})]\!]_\ell$ or $rec\mathcal{A}_n.t$ and has been generated by a derivation

$$\triangle(\triangle_1(\mathcal{N}_1^*) \parallel \cdots \parallel \triangle_i([\![\mathcal{A}_p(\widehat{u})]\!]_\ell) \parallel \cdots \parallel \triangle_n(\mathcal{N}_n^*)) \xrightarrow{rec} \triangle(\triangle_1(\mathcal{N}_1)^* \parallel \cdots \parallel \triangle_i([\![P\{\widehat{u}/\widehat{x}\}]\!]_\ell) \parallel \cdots \parallel \triangle_n(\mathcal{N}_n^*)$$

or

$$\triangle(\triangle_1(\mathcal{N}_1^*) \parallel \cdots \parallel \triangle_i(rec\mathcal{A}_n.t) \parallel \cdots \parallel \triangle_n(\mathcal{N}_n^*)) \xrightarrow{rec} \triangle(\triangle_1(\mathcal{N}_1^*) \parallel \cdots \parallel \triangle_i(t\{rec\mathcal{A}_n.t/\mathcal{A}_n\}) \parallel \cdots \parallel \triangle_n(\mathcal{N}_n^*))$$

such that for all $0 < j \leq n$ with $i \neq j$, $\mathcal{N}_j^* \rightarrow^* \mathcal{N}_j$ and $[\![P\{\widehat{u}/\widehat{x}\}]\!]_\ell \rightarrow^* \mathcal{N}_i$ or $t\{rec\mathcal{A}_n.t/\mathcal{A}_n\} \rightarrow^* \mathcal{N}_i$, since the black version of $\mathcal{A}_p(\widehat{u})$ or $rec\mathcal{A}_n.t$ can only occur in the scope of prefix, choice and recursion operators and no new parallel, communication merge, left merge, restriction, abstraction, and encapsulation operators are generated. Hence the term $\mathcal{N}' \in \varepsilon'_{\mathcal{M}}$ also occurs as a term $\triangle(\triangle_1(\mathcal{N}'_1) \parallel \cdots \parallel \triangle_i([\![\mathcal{A}_p(\widehat{u})]\!]_\ell) \parallel \cdots \parallel \triangle_n(\mathcal{N}'_n))$ or $\triangle(\triangle_1(\mathcal{N}'_1) \parallel \cdots \parallel \triangle_i(rec\mathcal{A}_n.t) \parallel \cdots \parallel \triangle_n(\mathcal{N}'_n))$ in $\varepsilon'_{\mathcal{M}}$ where $[\![\mathcal{A}_p(\widehat{u})]\!]_\ell$ or $rec\mathcal{A}_n.t$ is completely red. It follows that for each term $\mathcal{N} \in \varepsilon'_{\mathcal{M}}$ which is of the form $\triangle(\triangle_1(\mathcal{N}_1) \odot \cdots \odot \triangle_i(\mathcal{N}_i) \odot \cdots \odot \triangle_n(\mathcal{N}_n))$ where there is a $0 < j \leq n$ that $\mathcal{N}_j$ is completely black, there is a *maximum red term* $\mathcal{N}^* \in \varepsilon'_{\mathcal{M}}$ which is of the form $\triangle(\triangle_1(\mathcal{N}_1) \odot \cdots \odot \triangle_i(\mathcal{N}_i) \odot \cdots \odot \triangle_n(\mathcal{N}_n))$ where for all $0 < i \leq n$, $\mathcal{N}_i$ are partly/completely red while the length of its red part is maximal. Since each maximum red term is achieved from a maximum red term by a derivation during which the length of red part is reduced, then the number of maximum red terms is finite. It follows that $\varepsilon_{\mathcal{M}}$ is finite. □

F. Proof of the Case Study

We briefly explain how the equations in Section 7 can be proved by the application of *CNT* axioms. To this aim we use the following derived axioms:

$$C\eta.(rec\mathcal{A}_n.C'\tau.t + t) \overset{Fold,T_2,UnFold}{=} C\eta.(rec\mathcal{A}_n.t), \mathcal{A}_n \text{ is guarded in } t \qquad WUng_3$$

$$rec\mathcal{A}_n.C\tau.(\mathcal{A}_n + t) + C'\tau.(\mathcal{A}_n + t') + s \overset{WUng_{1-2}}{=} rec\mathcal{A}_n.C\tau.(\mathcal{A}_n + t + t') + s, C \subseteq C' \qquad WUng_4$$

By application of P_{2-5}, Bro, $LExe_{1-3}$, S_{1-4}, and $Sync_{1-5}$, it is straightforward to show that $\chi(0, ?, 0, 0, ?) = \chi(1, ?, 0, 0, ?)$, and we can derive the following equations.

$$\chi(0,?,0,0,?) = \{\}req!\{A\}.\chi(1,?,0,0,?) + \{\}req!\{A\}.\chi(1,?,0,3,?) + \{A \rightsquigarrow B\}req!\{A\}.\chi(1,?,1,0,?) + \{A \rightsquigarrow B\}req!\{A\}.\chi(1,?,1,3,?) + \{\}error(B)!\{A\}.\chi(0,?,0,0,?)$$

$$\chi(1,?,0,3,?) = \{\}req!\{A\}.\chi(1,?,0,3,?) + \{A \rightsquigarrow B\}req!\{A\}.\chi(1,?,1,3,?) + \{\}req!\{A\}.\chi(1,?,0,0,?) + \{\}req!\{A\}.\chi(1,?,1,0,?) + \{\}error(B)!\{A\}.\chi(1,?,0,3,?)$$

$$\chi(1,?,1,0,?) = \{\}req!\{A\}.\chi(1,?,1,0,?) + \{\}req!\{A\}.\chi(1,?,1,3,?) + \{\}rep(B)!\{B\}.\chi(1,?,0,0,?) + \{B \rightsquigarrow A\}rep(B)!\{B\}.\chi(0,B,0,0,?) + \{B \rightsquigarrow A\}rep(B)!\{A\}.\chi(0,B,0,4,?) + \{\}rep(B)!\{A\}.\chi(1,?,0,4,?)$$

$$\chi(0,B,0,0,?) = \{\}data(B)!\{A\}.\chi(0,B,0,0,?) + \{A \rightsquigarrow B\}data(B)!\{A\}.\chi(0,B,0,0,?) + \{\}error(B)!\{B\}.\chi(0,B,0,0,?) + \{B \rightsquigarrow A\}error(B)!\{B\}.\chi(0,?,0,0,?)$$

For each χ with a parameter setting, we can derive an equation as above. Since all equations are guarded, by *Fold* we can derive $\chi(\theta)$ is a solution for Z_θ, where θ is the sequence of parameter values. Recall that $Z \equiv Z_{0,?,0,0,?}$, $Z_B \equiv Z_{0,B,0,0,?}$, and $Z_C \equiv Z_{0,C,0,0,B}$. Thus by *Con* and *Fold*, $\chi(0,B,0,0,?)$ is a solution for the following recursive equation:

$$\chi(0,B,0,0,?) = recZ_B.\{\}data(B)!\{A\}.Z_B + \{\}error(B)!\{B\}.Z_B + \{B \rightsquigarrow A\}error(B)!\{B\}.\chi(0,?,0,0,?).$$

Then by *Hid* and Abs_{1-3}:

$$\nabla_{\{req,rep,error\}}(\chi(0,B,0,0,?)) = \nabla_{\{req,rep,error\}}(recZ_B.\{\}data(B)!\{A\}.Z_B + \{\}\tau.Z_B + \{B \rightsquigarrow A\}\tau.\chi(0,?,0,0,?)) \overset{WUng_2}{\Rightarrow}$$
$$\nabla_{\{req,rep,error\}}(\chi(0,B,0,0,?)) = \nabla_{\{req,rep,error\}}(recZ_B.\{\}\tau.(\{\}data(B)!\{A\}.Z_B + \{B \rightsquigarrow A\}\tau.\chi(0,?,0,0,?)) + \{\}data(B)!\{A\}.Z_B + \{A \rightsquigarrow B\}\tau.\chi(0,?,0,0,?)).$$

Then by derived axiom $WUng_3$ the following equation holds:

$$C\eta.\nabla_{\{req,rep,error\}}(\chi(0,B,0,0,?)) = C.\eta\nabla_{\{req,rep,error\}}(recZ_B.\{\}data(B)!\{A\}.Z_B + \{B \rightsquigarrow A\}\tau.\chi(0,?,0,0,?)).$$

Similarly the equation for $\chi(0,C,0,0,B)$ can be derived, as explained in Section 7. By repeating the argumentation above, the following equations can be derived:

$$\nabla_M(\chi(0,?,0,0,?)) = recZ.\{\}\tau.Z + \{\}\tau.\nabla_M(\chi(1,?,0,3,?)) + \{\}\tau.\nabla_M(\chi(1,?,1,0,?)) + \{\}\tau.\nabla_M(\chi(1,?,1,3,?))$$
$$\{\}\tau.\nabla_M(\chi(1,?,0,3,?)) = \{\}\tau.recZ_{1,?,0,3,0}.\{A \rightsquigarrow B\}\tau.\nabla_M(\chi(1,?,1,3,?)) + \{\}\tau.Z + \{\}\tau.\nabla_M(\chi(1,?,1,0,?))$$
$$\{\}\tau.\nabla_M(\chi(1,?,1,3,?)) = \{\}\tau.recZ_{1,?,1,3,?}.\{\}\tau.\nabla_M(\chi(1,?,1,0,?)) + \{\}\tau.\nabla_M(\chi(1,?,0,3,?)) + \{B \rightsquigarrow A\}\tau.\nabla_M(\chi(0,B,0,3,?))$$

where $M = \{req, rep, error\}$. By application of $UnFold$ and equation $\{\}\tau.\nabla_M(\chi(0,B,0,0,?)) = \{\}\tau.\nabla_M(\chi(0,B,0,3,?))$ (which is straightforward to prove), the following equation holds:

$$\nabla_M(\chi(0,?,0,0,?)) = recZ.\{\}\tau.Z + \{\}\tau.\nabla_M(\chi(1,?,1,0,?)) + \{\}\tau.\nabla_M(\chi(B,0,0,0,?)).$$

By the equation of $\{\}\tau.\nabla_M(\chi(1,?,1,0,?))$, and $UnFold$:

$$l\nabla_M(\chi(0,?,0,0,?)) = recZ.\{\}\tau.Z + \{\}\tau.\nabla_M(\chi(C,0,0,0,B)) + \{\}\tau.\nabla_M(\chi(B,0,0,0,?)). \quad (5)$$

where the unguarded Z can be removed by $WUng_2$. Finally, by the equation of $\nabla_M(\chi(B,0,0,0,?))$, $UnFold$, T_2, and $Fold$:

$$\nabla_M(\chi(0,?,0,0,?)) = recZ.\{\}\tau.\nabla_M(\chi(C,0,0,0,B)) + \{\}\tau.\nabla_M(\chi(B,0,0,0,?)).$$

By substituting the recursions for $\{\}\tau.\nabla_M(\chi(B,0,0,0,?))$ and $\{\}\tau.\nabla_M(\chi(C,0,0,0,B))$ in equation 5, $UnFold$, Hid, and $WUng_{1,2}$:

$$\begin{aligned}
&\nabla_M(\chi(0,?,0,0,?)) = \\
&\quad recZ.\{\}\tau.(Z + \{\}data(B)!\{A\}.\nabla_M(\chi(B,0,0,0,?))) + \{\}\tau.Z + \\
&\qquad \{\}\tau.(Z + \{\}data(?)!\{A\}.\nabla_M(\chi(C,0,0,0,B)) + \\
&\qquad\quad data(?)!\{A\}.\nabla_M(recZ_{CB}.(\{\}data(?)!\{A\}.Z_{CB} + \{\}data(B)!\{?\}.\chi(C,0,0,0,B))) + \\
&\qquad\quad \{\}data(?)!\{A\}.\nabla_M(\chi(C,0,0,2,B)) + \\
&\qquad\quad \{\}\tau.\nabla_M(recZ_d.\{\}data(?)!\{A\}.Z_d))
\end{aligned}$$

where $\{\}\tau.\nabla_M(\chi(C,0,0,2,B)) = \{\}\tau.recZ_e.(\{\}data(?)!\{A\}.Z_e + \{? \rightsquigarrow A\}\tau.Z + \{\}\tau.recZ_d.\{\}data(?)!\{A\}.Z_d)$. By the derived axiom $WUng_4$ and $WUng_2$:

$$\begin{aligned}
&\nabla_M(\chi(0,?,0,0,?)) = \\
&\quad \nabla_M(recZ.\{\}\tau.(\\
&\qquad \{\}data(B)!\{A\}.\chi(B,0,0,0,?) + \\
&\qquad \{\}data(?)!\{A\}.\chi(C,0,0,0,B) + \\
&\qquad \{\}data(?)!\{A\}.recZ_{CB}.(\{\}data(?)!\{A\}.Z_{CB} + \{\}data(B)!\{?\}.\chi(C,0,0,0,B)) + \\
&\qquad \{\}data(?)!\{A\}.\chi(C,0,0,2,B) + \\
&\qquad \{\}\tau.recZ_d.\{\}data(?)!\{A\}.Z_d)).
\end{aligned}$$

By equations 1, 2 and 3, $C\eta.\nabla_M(\chi(0,?,0,0,?)) = C\eta.\nabla_M(\chi(B,0,0,0,?))$, $C\eta.\nabla_M(\chi(0,?,0,0,?)) = C\eta.\nabla_M(\chi(C,0,0,0,B))$, and $C\eta.\nabla_M(\chi(C,0,0,0,B)) = C\eta.\nabla_M(\chi(C,0,0,2,B))$ hold. Thus by Hid, $UnFold$, the equalities above and recursion, equation 4 in Section 7 has been derived.

G. Ground-Completeness of the *CNT* Axiomatization

We are going to prove Theorem 8.1, that the axiomatization of *CNT* is ground-complete for closed, finite-state terms modulo rooted branching computed network bisimilarity. To this aim, we perform the following steps:

1. first we show that each CNT_f term can be turned into a *normal form* consisting of only $0, C\eta.t', t' + t''$ and $rec\mathcal{A}_n.t'$, which is guarded;

2. next we define *recursive network specification* and prove that each guarded recursive network specification has a unique solution;

3. finally we show that our axiomatization is ground-complete for normal forms, by showing that equivalent normal forms are solutions for the same guarded recursive network specification.

Completeness of our axiomatization for all CNT_f terms results from steps 1 and 3. In the following sections we go through the above-mentioned steps.

G.1. Normal Forms

Definition G.1. *Normal forms* are terms made up of only $0, C\eta.t'$, $t' + t''$, and $rec\mathcal{A}_n.t'$, where $\mathcal{A}_n$ is guarded in t'.

Lemma G.1. Any normal form t can be turned by the axiomatization in Table 4 into a so-called *head normal form* $\sum\{C\eta.t^* | t \xrightarrow{\eta}_C t^*\}$.

Proof:
We prove this by induction on the maximal length of the inference tree by which $C_i\eta_i$ transitions of t are inferred. The base cases of the induction, $t \equiv 0$ or $t \equiv C\eta.t'$ are trivial. The inductive cases are the following ones:

- if $t \equiv t' + t''$, then t can be turned into the desired form by just summing the terms obtained by applying the inductive argument to t' and t''.

- if $t \equiv rec\mathcal{A}_n.t'$, then t can be turned into the desired form by directly considering the term obtained by applying the inductive argument to $t'\{rec\mathcal{A}_n.t'/\mathcal{A}_n\}$; by $Unfold$, $t = t'\{rec\mathcal{A}_n.t'/\mathcal{A}_n\}$, and by the operational rule Rec its transitions are those achieved by $t'\{rec\mathcal{A}_n.t'/\mathcal{A}_n\}$.

□

Lemma G.2. Let t, t' be normal forms. Then $t'' \equiv t \parallel t'$ or $t \mathbb{L} t'$ or $t \mid t'$ or $(\nu\ell)t$ or $\nabla_M(t)$ or $\partial_M(t)$ can be turned by the axiomatization in Table 4 into a head normal form $\sum\{C\eta.t^* | t'' \xrightarrow{\eta}_C t^*\}$.

Proof:
straightforward. □

Proposition G.1. Given the data model $\mathbb{D}$ with finite data domains, each closed term t of CNT_f whose bound protocol names have finite-state model and whose network names do not occur in the scope of one of the operators $\parallel, \mathbb{L}, |, (\nu\ell), \nabla_M$ or ∂_M for some $\ell \in Loc$ and $M \subseteq Msg$, can be turned into a normal form.

Proof:
We prove this by structural induction over the syntax of terms t (possibly open). The base cases of induction for $t \equiv 0$ or $t \equiv \mathcal{A}_n$ are trivial because they are in normal form already.

The inductive cases of the induction are the following ones:

- if $t \equiv [\![0]\!]_\ell$, then by application of P_1 we have $t = 0$, which is in normal form.

- if $t \equiv [\![\alpha.P']\!]_\ell$ or $t \equiv [\![P' + P'']\!]_\ell$ or $t \equiv [\![[u_1 = u_2]P', P'']\!]_\ell$, then t can be turned into a normal form by application of axioms $Pr_{4,5}$, $P_{0,2-5}$ and induction over $[\![P']\!]_\ell$ and $[\![P'']\!]_\ell$.

- if $t \equiv C\eta.t'$ or $t \equiv t' + t''$, then t can be turned into normal form by induction over t' and t''.

- if $t \equiv t' \parallel t''$ or $t \equiv t' \between t''$ or $t \equiv t' \mid t''$ or $t \equiv (\nu\ell)t'$ or $t \equiv \partial_M(t')$ for some $\ell \in Loc$ and $M \in Msg$, then following the approach of [1], we can turn t into normal form as follows. By induction over t' and t'', we first obtain t''' by replacing t' and t'' inside t by their corresponding normal forms. Since t has a finite-state transition system by Proposition 8.1, t''' has a finite-state transition system. Let $t_1, \cdots, t_n$ be the states of the transition system of t''' with $t''' \equiv t_n$. It can be easily seen that, due to Lemma G.2, there exist m_i, $\{C^i_j\}_{j \leq m_i}$ (denoting network restrictions), $\{\eta^i_j\}_{j \leq m_i}$ (denoting actions), $\{k^i_j\}_{j \leq m_i}$ (denoting natural numbers) such that we can derive $t_i = \sum_{j \leq m_i} C^i_j \eta^i_j.t_{k^i_j}$. Hence we can characterize the behavior of t''' by means of a recursive operators $rec\mathcal{A}_{n_n}.t_{\mathcal{A}_{n_n}}$ such that $t''' \equiv t_n$ is the answer of such recursion. So we can turn t''' to a normal form t'''' as follows. For each i from 1 to n, we do the following: if i is such that $\exists j \leq m_i \cdot k^i_j = i$, by application of $Fold$ we have $t_i = rec\mathcal{A}_{n_i}.(\sum_{j \leq m_i \cdot k^i_j \neq i} C^i_j \eta^i_j.t_{k^i_j} + \sum_{j \leq m_i \cdot k^i_j = i} C^i_j \eta^i_j.\mathcal{A}_{n_i})$. It should be noted that axiom $Fold$ is applicable, since t' and t'' have been turned in to normal forms and contain guarded recursions only, hence (since the operators considered cannot turn visible actions into τ) every cycle in the derived transition system contains at least a visible action. Then replace each subterm $t_{i+1}, \cdots, t_n$ with its equivalent recursion. When we have replaced t_{n-1} in $t_n \equiv t'''$, we are done.

- if $t \equiv rec\mathcal{A}_n.t'$, then following the approach of [8], we show by induction on the depth of nesting of recursions in t' that there exists a guarded term t'' such that:

  - $\mathcal{A}_n$ is guarded in t'';
  - no free unguarded occurrence of any network name in t'' lies within a recursion in t''; and
  - $UnFold, Ung, WUng_{1,2} \vdash rec\mathcal{A}_n.t' = rec\mathcal{A}_n.t''$.

  Assume that this property holds for every s whose recursion depth is less than that of t'. Then for each recursion $rec\mathcal{A}_m.s$ in t' that lies within no recursion in t', there must be a guarded term s' such that $\mathcal{A}_m$ is guarded in s', no free unguarded occurrence of any network name in s' lies within a recursion in s', and $UnFold, Ung, WUng_{1,2} \vdash rec\mathcal{A}_m.s = rec\mathcal{A}_m.s'$.

  Let t''' be the term resulting from simultaneously replacing every such top level $rec\mathcal{A}_m.s$ in t' by $s'\{rec\mathcal{A}_m.s'/\mathcal{A}_m\}$. Since t' is essentially finite-state, clearly t''' is guarded and no free unguarded occurrence of any network name in t''' lies within a recursion, parallel , left merge, restriction, abstraction, and encapsulation operator in t'''. Now we remove unguarded occurrences of $\mathcal{A}_n$ in $rec\mathcal{A}_n.t'''$, knowing that they do not lie within recursions, by application of $Ung, WUng_{1,2}$.

- if $t \equiv [\![\mathcal{A}_p(\widehat{u})]\!]_\ell$, then by assigning a fresh network name, represented by $\mathcal{A}_{p(\widehat{u})}$, to each $[\![\mathcal{A}_p(\widehat{u})]\!]_\ell$ and application of axioms $P_{0,2-5}$ and $Unfold$, it can be turned into a recursive term $t = rec\mathcal{A}_{p(u)}.t'$, where t' may contain terms of the form $[\![\mathcal{A}_q(\widehat{u'})]\!]_\ell$. We repeat this process for each free $[\![\mathcal{A}_q(\widehat{u'})]\!]_\ell$, i.e. not in the scope of $rec\mathcal{A}_{q(\widehat{u'})}$, until no such a term remains in t'. During this process we should

substitute each bound $[\![\mathcal{A}_q(\widehat{u}')]\!]_\ell$, i.e. occurred in the scope of $rec\mathcal{A}_{q(\widehat{u}')}$ by its corresponding network name. This process terminates since the number of protocol names is finite and the data model contains finite data domains, so the number of protocol name instances is finite.

- if $t \equiv \nabla_M(t')$ then t is turned into a normal form as follows. First by application of induction over t', it can be turned into a normal form t''. It should be noted that t'' cannot include free network names and has a finite-state transition system.

  First we show by structural induction that for any normal form t'', $\nabla_M(t'')$ can be turned into $\nabla_M(t''')$, where t''' obtained from t'' by syntactically replacing each occurrence of action η by $\delta_M(\eta)$. The base cases of induction for $t'' \equiv 0$ or $t'' \equiv \mathcal{A}_n$ are trivial. The inductive cases of the induction are the following ones:

  - if $t'' \equiv C\eta.t_1''$, then by application of Abs_1, $t''' = \nabla_M(t'') = C\delta_M(\eta).\nabla_M(t_1'')$, which by induction can be turned into $= C\delta_M(\eta).\nabla_M(t_1''')$, such that in t_1''' each occurrence of action η is replaced by $\delta_M(\eta)$. Finally $\nabla_M(t''')$ is $\nabla_M(C\delta_M(\eta).t_1''')$.
  - if $t'' \equiv t_1'' + t_2''$, then by application of Abs_2, $\nabla_M(t'')$ can be turned into $\nabla_M(t_1'') + \nabla_M(t_2'')$, which by induction can be turned into $\nabla_M(t_1''') + \nabla_M(t_2''')$ such that in t_1''' and t_2''' each occurrence of action η is replaced by $\delta_M(\eta)$. Finally we obtain $\nabla_M(t''')$ as $\nabla_M(t_1''' + t_2''')$.
  - if $t'' \equiv rec\mathcal{A}_n.t_1''$, then by application of axiom Hid, $\nabla_M(t'')$ can be turned into $rec\mathcal{A}_n.\nabla_M(t_1'')$, which by induction can be turned into $rec\mathcal{A}_n.\nabla_M(t_1''')$ such that in t_1''' each occurrence of action η is replaced by $\delta_M(\eta)$. Finally we obtain $\nabla_M(t''')$ by application of Hid again as $\nabla_M(rec\mathcal{A}_n.t_1''')$.

  Notice that, due to the usage of axiom Hid in the last item, the equational transformation procedure from $\nabla_M(t'')$ to $\nabla_M(t''')$ arising from the above induction works on CNT.

  Then we use Ung and $WUng_{1,2}$ to get rid of generated unguarded recursion into t''' to get a guarded t''''. Finally we consider $\nabla_M(t'''')$ and we apply the same technique as for, e.g., the $\|$ operator to turn it into normal form (exploiting the fact that t'''' is guarded, finite-state and does not include free network names). In particular now we can do that because the application of the abstraction operator has no effect on labels of transitions, hence it cannot generate cycles made up of only τ actions when the semantics is considered.

□

G.2. Recursive Network Specification

Definition G.2. [8] A recursive network specification $E = E(S)$ is a set of equations $E = \{\mathcal{A}_n = t_{\mathcal{A}_n} | \mathcal{A}_n \in S\}$, where $t_{\mathcal{A}_n}$ is a term over the CNT_f signature and names from S where $S \subseteq \mathcal{A}_N$. $\mathcal{N} \in CNT_f$ provably satisfies the recursive network specification E in $\mathcal{A}_{n_0} \in S$ if there are terms $\mathcal{N}_{\mathcal{A}_n}$ for $\mathcal{A}_n \in S$ with $\mathcal{N} = \mathcal{N}_{\mathcal{A}_{n_0}}$, such that for each $\mathcal{A}_n \in S$, $\mathcal{N}_{\mathcal{A}_n} = t_{\mathcal{A}_n}\{\mathcal{N}_{\mathcal{A}_n'}/\mathcal{A}_n'\}_{\mathcal{A}_n' \in S}$.

In the above definition $t_{\mathcal{A}_n}\{t'_{\mathcal{A}_n'}/\mathcal{A}_n'\}_{\mathcal{A}_n' \in S}$ denotes the result of simultaneously replacing $t'_{\mathcal{A}_n'}$ for each $\mathcal{A}_n' \in S$. We call a recursive network specification guarded if all occurrences of all its network names in the right-hand sides of all its definitions are guarded or it can be rewritten to such a network recursive specification using the axioms of the theory and the equations of the specification:

Definition G.3. Let $E = E(S)$ be a recursive network specification. The relation $\longrightarrow \subseteq S \times S$ is defined as $\mathcal{A}_n \longrightarrow \mathcal{A}'_n$ if $\mathcal{A}'_n$ occurs unguarded in the right-hand side equation defining $\mathcal{A}_n$. Now E is called guarded if $\longrightarrow$ is well-founded.

Proposition G.2. If $E = E(S)$ is a finite guarded recursive network specification and $\mathcal{A}_{n_0} \in S$, then there is a closed term in CNT_f which provably satisfies E in $\mathcal{A}_{n_0}$. Moreover, if there are two such terms $\mathcal{N}_1$ and $\mathcal{N}_2$, then $Fold \vdash \mathcal{N}_1 = \mathcal{N}_2$.

Proof:
We can consider each $C\eta$ prefix as a new action (or τ, if their composition will not make a network name guarded), so each $C\eta.\mathcal{N}$ term can be considered as a prefix term in *CCS*. Consequently the proof of the above proposition follows from Proposition G.1; because normal forms are like *CCS* terms, and the above Proposition over *CCS* terms has been proved in [23]. □

G.3. Completeness of the CNT_f-Axiomatization

Theorem G.1. Let $\mathcal{N}_1$ and $\mathcal{N}_2$ be two normal terms in CNT_f such that $\mathcal{N}_1 \simeq_{rb} \mathcal{N}_2$. Then given a data model $\mathbb{D}$ with finite data domains, there is a finite recursive network specification $E = E(S)$ provably satisfied in the same variable $\mathcal{A}_{n_0} \in S$ by both $\mathcal{N}_1$ and $\mathcal{N}_2$.

Proof:
Take a fresh set of network names $S = \{\mathcal{A}_{n_{\mathcal{N}'_1\mathcal{N}'_2}} | \mathcal{N}'_1 \in \varepsilon_{\mathcal{N}_1}, \mathcal{N}'_2 \in \varepsilon_{\mathcal{N}_2}, \mathcal{N}'_1 \simeq_b \mathcal{N}'_2\}$ (which is finite by Proposition 8.1). Let $\mathcal{A}_{n_0} = \mathcal{A}_{n_{\mathcal{N}_1\mathcal{N}_2}}$. Let η_τ range over the set of receive actions and τ. Now for each $\mathcal{A}_{n_{\mathcal{N}'_1\mathcal{N}'_2}} \in S$, E contains the equation

$$\begin{aligned}\mathcal{A}_{n_{\mathcal{N}'_1\mathcal{N}'_2}} = &\sum\{C\eta\mathcal{A}_{n_{\mathcal{N}''_1\mathcal{N}''_2}} | \mathcal{N}'_1 \xrightarrow{\eta}_C \mathcal{N}''_1, \mathcal{N}'_2 \xrightarrow{\langle\eta\rangle}_C \mathcal{N}''_2, \mathcal{N}''_1 \simeq_b \mathcal{N}''_2\} + \\ &\sum\{C\eta_\tau\mathcal{A}_{n_{\mathcal{N}''_1\mathcal{N}'_2}} | \mathcal{A}_{n_{\mathcal{N}'_1\mathcal{N}'_2}} \neq \mathcal{A}_{n_0}, \mathcal{N}'_1 \xrightarrow{\eta_\tau}_C \mathcal{N}''_1, \mathcal{N}''_1 \simeq_b \mathcal{N}'_2\} + \\ &\sum\{C\eta_\tau\mathcal{A}_{n_{\mathcal{N}'_1\mathcal{N}''_2}} | \mathcal{A}_{n_{\mathcal{N}'_1\mathcal{N}'_2}} \neq \mathcal{A}_{n_0}, \mathcal{N}'_2 \xrightarrow{\eta_\tau}_C \mathcal{N}''_2, \mathcal{N}'_1 \simeq_b \mathcal{N}''_2\}\end{aligned}$$

The recursive network specification $E = E(S)$ is guarded, using that $\mathcal{A}_{n_{\mathcal{N}'_1\mathcal{N}'_2}} \longrightarrow \mathcal{A}_{n_{\mathcal{N}''_1\mathcal{N}''_2}}$ iff $\mathcal{A}_{n_{\mathcal{N}'_1\mathcal{N}'_2}}$ has a summand $C\tau.\mathcal{A}_{n_{\mathcal{N}''_1\mathcal{N}''_2}}$. It is easy to show that any infinite $\mathcal{A}_{n_{\mathcal{N}'_1\mathcal{N}'_2}} \longrightarrow \mathcal{A}_{n_{\mathcal{N}''_1\mathcal{N}''_2}} \longrightarrow \cdots$ implies an infinite path of performing τ $\mathcal{N}'_1 \xrightarrow{\tau}_{C'} \mathcal{N}''_1 \xrightarrow{\tau}_{C''} \cdots$ which cannot exist since $\mathcal{N}_1$ and $\mathcal{N}_2$ are closed normal forms. It must be shown that $\mathcal{N}_1$, provably satisfies E in $\mathcal{A}_{n_0}$. The same statement for $\mathcal{N}_2$ follows by symmetry.

For $\mathcal{A}_{n_{\mathcal{N}'_1\mathcal{N}'_2}} \in S$, let $\mathcal{N}^*_{\mathcal{N}'_1\mathcal{N}'_2}$ be the term

$$\begin{aligned}&\sum\{C\eta.\mathcal{N}''_1 | \mathcal{N}'_1 \xrightarrow{\eta}_C \mathcal{N}''_1, \mathcal{N}'_2 \xrightarrow{\langle\eta\rangle}_C \mathcal{N}''_2, \mathcal{N}''_1 \simeq_b \mathcal{N}''_2\} + \\ &\sum\{C\eta_\tau\mathcal{N}''_1 | \mathcal{A}_{n_{\mathcal{N}'_1\mathcal{N}'_2}} \neq \mathcal{A}_{n_0}, \mathcal{N}'_1 \xrightarrow{\eta_\tau}_C \mathcal{N}''_1, \mathcal{N}''_1 \simeq_b \mathcal{N}'_2\}\end{aligned}$$

and define the term $\mathcal{M}^*_{\mathcal{N}'_1\mathcal{N}'_2}$ as

$$\begin{cases} \mathcal{N}^*_{\mathcal{N}'_1\mathcal{N}'_2} + C\eta_\tau.\mathcal{N}'_1 & \mathcal{A}_{n_{\mathcal{N}'_1\mathcal{N}'_2}} \neq \mathcal{A}_{n_0}, \exists\mathcal{N}''_2 \cdot \mathcal{N}'_2 \xrightarrow{\eta_\tau}_C \mathcal{N}''_2, \mathcal{N}'_1 \simeq_b \mathcal{N}''_2 \\ \mathcal{N}'_1 & \text{otherwise} \end{cases}$$

It follows from Lemma G.1 that $\mathcal{N}_1' = \mathcal{N}_1' + \mathcal{N}^*_{\mathcal{N}_1'\mathcal{N}_2'}$ and hence $C\eta.(\mathcal{N}^*_{\mathcal{N}_1'\mathcal{N}_2} + C'\tau.\mathcal{N}_1') =^{T_2} C\eta.\mathcal{N}_1'$. Thus

$$C\eta.\mathcal{M}^*_{\mathcal{N}_1'\mathcal{N}_2'} = C\eta.\mathcal{N}_1' \tag{6}$$

It suffices to prove that for $\mathcal{A}_{n_{\mathcal{N}_1'\mathcal{N}_2'}} \in S$

$$\begin{aligned}\mathcal{M}^*_{\mathcal{N}_1'\mathcal{N}_2'} = &\sum\{C\eta\mathcal{M}^*_{\mathcal{N}_1''\mathcal{N}_2''} | \mathcal{N}_1' \xrightarrow{\eta}_C \mathcal{N}_1'', \mathcal{N}_2' \xrightarrow{\langle\eta\rangle}_C \mathcal{N}_2'', \mathcal{N}_1'' \simeq_b \mathcal{N}_2''\} + \\ &\sum\{C\eta_\tau\mathcal{M}^*_{\mathcal{N}_1''\mathcal{N}_2'} | \mathcal{A}_{n_{\mathcal{N}_1'\mathcal{N}_2'}} \neq \mathcal{A}_{n_0}, \mathcal{N}_1' \xrightarrow{\eta_\tau}_C \mathcal{N}_1'', \mathcal{N}_1'' \simeq_b \mathcal{N}_2'\} + \\ &\sum\{C\eta_\tau\mathcal{M}^*_{\mathcal{N}_1'\mathcal{N}_2''} | \mathcal{A}_{n_{\mathcal{N}_1'\mathcal{N}_2'}} \neq \mathcal{A}_{n_0}, \mathcal{N}_2' \xrightarrow{\eta_\tau}_C \mathcal{N}_2'', \mathcal{N}_1' \simeq_b \mathcal{N}_2''\}.\end{aligned}$$

By equation 6, this is equivalent to

$$\mathcal{M}^*_{\mathcal{N}_1'\mathcal{N}_2'} = \mathcal{N}^*_{\mathcal{N}_1'\mathcal{N}_2'} + \sum\{C\eta_\tau.\mathcal{N}_1' | \mathcal{A}_{n_{\mathcal{N}_1'\mathcal{N}_2'}} \neq \mathcal{A}_{n_0}, \mathcal{N}_2' \xrightarrow{\eta_\tau}_C \mathcal{N}_2'', \mathcal{N}_1' \simeq_b \mathcal{N}_2''\}. \tag{7}$$

There are three cases to examine:

- If $\mathcal{A}_{n_{\mathcal{N}_1'\mathcal{N}_2'}} \neq \mathcal{A}_{n_0}$, and $\exists\mathcal{N}_2''\cdot\mathcal{N}_2' \xrightarrow{\eta_\tau}_C \mathcal{N}_2'', \mathcal{N}_1' \simeq_b \mathcal{N}_2''$, this follows from the definition of $\mathcal{M}^*_{\mathcal{N}_1'\mathcal{N}_2'}$.
- If $\mathcal{A}_{n_{\mathcal{N}_1'\mathcal{N}_2'}} \neq \mathcal{A}_{n_0}$, and $\nexists\mathcal{N}_2''\cdot\mathcal{N}_2' \xrightarrow{\eta_\tau}_C \mathcal{N}_2'', \mathcal{N}_1' \simeq_b \mathcal{N}_2''$, equation 7 reduces to $\mathcal{N}_1' = \mathcal{N}^*_{\mathcal{N}_1'\mathcal{N}_2'}$, and by Lemma G.1 if suffices to establish that "if $\mathcal{N}_1' \xrightarrow{\eta}_C \mathcal{N}_1''$ then η is of the form $m(\widehat{u})?$ or τ and $\mathcal{N}_1'' \simeq_b \mathcal{N}_2'$ or $\exists\mathcal{N}_2''\cdot\mathcal{N}_2' \xrightarrow{\langle\eta\rangle}_C \mathcal{N}_2'' \wedge \mathcal{N}_1'' \simeq_b \mathcal{N}_2''$". This follows from the fact that $\mathcal{N}_1' \simeq_b \mathcal{N}_2'$, and if $\mathcal{N}_2 \xrightarrow{\tau}_{C'} \mathcal{N}_2''' \Rightarrow \mathcal{N}_2''$ such that $\mathcal{N}_2'' \simeq_b \mathcal{N}_1' \simeq_b \mathcal{N}_2'$, then by Lemma A.2 $\mathcal{N}_1' \simeq_b \mathcal{N}_2'''$, violating the assumption.
- If $\mathcal{A}_{n_{\mathcal{N}_1'\mathcal{N}_2'}} = \mathcal{A}_{n_0}$, then equation 7 reduces to $\mathcal{N}_1' = \mathcal{N}^*_{\mathcal{N}_1'\mathcal{N}_2'}$, and we should show that "if $\mathcal{N}_1' \xrightarrow{\eta}_C \mathcal{N}_1''$, then $\exists\mathcal{N}_2''\cdot\mathcal{N}_2' \xrightarrow{\langle\eta\rangle}_C \mathcal{N}_2'' \wedge \mathcal{N}_1'' \simeq_b \mathcal{N}_2''$", which follows immediately from $\mathcal{N}_1' \simeq_{rb} \mathcal{N}_2'$.

□

Corollary G.1. Let $\mathcal{N}_1$ and $\mathcal{N}_2$ be two normal forms in *CNT*$_f$ such that, given a data domain $\mathbb{D}$ with finite data domains, $\mathcal{N}_1 \simeq_{rb} \mathcal{N}_2$. Then $\mathcal{N}_1 = \mathcal{N}_2$.

Theorem 8.1 follows from Proposition G.1 and the above Corollary. Suppose $\mathcal{N}_1 \simeq_{rb} \mathcal{N}_2$. Using Proposition G.1, there are two normal forms $\mathcal{N}_1'$ and $\mathcal{N}_2'$ (which are guarded) such that $\mathcal{N}_1 = \mathcal{N}_1'$ and $\mathcal{N}_2 = \mathcal{N}_2'$. Soundness of the axiomatization yields $\mathcal{N}_1 \simeq_{rb} \mathcal{N}_1'$ and $\mathcal{N}_2 \simeq_{rb} \mathcal{N}_2'$. Transitivity of rooted branching computed network bisimilarity yields $\mathcal{N}_1' \simeq_{rb} \mathcal{N}_2'$, and by application of Corollary G.1, $\mathcal{N}_1' = \mathcal{N}_2'$ and consequently $\mathcal{N}_1 = \mathcal{N}_2$.