Verifying the Arrow Distributed Protocol in a Spatial Logic

A. Ribeiro L. Caires L. Monteiro

Departamento de Informática
DI/FCT/UNL
Portugal

July 2004

Abstract

We develop, in a spatial logic, a specification and verification of a distributed peer-2-peer protocol, the Arrow Distributed Directory Protocol of Demmer and Herlihy. The Arrow Protocol is a algorithm for directory lookup, allowing any member node of the directory to get access to an object, for example an MP3 file, stored at some other node. Our specification addresses both behavioral and structural aspects of the directory, for instance, when specifying a node's behavior, or the linked structure of the directory. We also exemplify how the logic can be used to prove (using formal logical reasoning, not model-checking) some safety properties essential for establishing the correctness of the protocol, such as the property that the graph underlying the directory will never contain cycles or alternative paths between any pair of nodes, and that the path from every node will always reach a node that has or will eventually acquire ownership of the object.

Contents

1 Introduction

Spatial logics for concurrency [1, 2, 3, 4, 5] support the specification not only of behavioral properties but also of structural properties of concurrent systems, revealing adequate to formally describe and reason about distributed systems. More traditional behavioral logics, such as variants of Hennessy-Milner logic [6, 9], are intended to describe the interactive behavior and evolution of a system in time, not in space. Spatial logics introduce novel spatial operators and combine them with the behavioral operators such as the diamond (one-step) and box (all-steps) operators for a one-step transition, and the minimal and maximal fix-point operators, for a possibly finite number of steps. By combining spatial with behavioral operators, we may express spatial properties of a system's state after some finite or infinite number of transitions, and then, its evolution in space as well as in time. For some examples of spatial properties consider: the number of components a system has, what properties are satisfied by each of them individually, or by some part of the system, how the system's structure (e.g., the access paths between components) evolve in the course of computation.

In this work, we use the spatial logic of Caires and Cardelli [1, 2, 3] to develop a case study on the specification and verification of behavioral and spatial properties a distributed systems. We consider the *Arrow Distributed Directory Protocol* of Demmer and Herlihy [7], a distributed peer-2-peer protocol for directory lookup. The Arrow Protocol is a algorithm for directory lookup, allowing any member node of the directory to get access to an object (for example, a MP3 file) stored at some other node. The algorithm is executed by a distributed network of nodes, where at each moment the structure of the access paths between nodes determine who the messages corresponding to the requests for the ownership of an object, located at a single node, are propagated.

In the preliminary work described in this report, we access the usefullness of a spatial logic, using its expressive power in an essential way to formally prove the main safety property of the protocol: the property that the path followed by a request always ends on a terminal node, so every node that requests the object will eventually get it. Properties of this kind are not expressible in standard behavioral logics, because such logics are unable to describe properties of the structure of a closed system such as the one we are considering. We then develop a general approach to prove safety properties for a distributed system, given the specification of the behavior of each atomic component, and a specification of the structure of the initial state. In our example, the property is fairly general, in the sense that neither the number of nodes nor the connections between them is, a priori, determined, so that our proof applies to any directory with an arbitrary number of nodes. To the specification which are specific to our case study, we also add some domain specific axioms, stating general properties of our intended model, which is the asynchronous π-calculus (see [8]).

During the development, we need to express many properties of the underlying graph of the network (the graph whose vertices are the nodes in the network and whose edges are the connections and the messages in transit between them). They are mostly spatial properties, such as the property that states the uniqueness of the undirected path between any pair of nodes. As a consequence of the uniqueness of every path in this undirected graph, we may conclude there are no cycles in the directed graph whose edges correspond to physical connections between the nodes, and consequently, the path that starts on every node and leads to a terminal one becomes well-defined.

We also prove three invariants for the system. The simplest expresses that the unitary components of the system are the nodes and the messages. Secondly, we prove that the undirected graph mentioned before has indeed the property of uniqueness of paths. Finally, at any time, and for every node in the graph, there is always a unique directed path leading to a terminal node, which can be the owner, or a node that has already requested the object and is still waiting for it.

2 The Spatial Logic and the Asynchronous π-calculus

In this section, we briefly review the basic spatial logic of [1, 2, 3].

Definition 2.1 (The Spatial Logic)

$\mathbf{0}$	*Void*
$\mathbf{F}$	*False*
$m = n$	*Equality*
$m\langle \ldots \rangle$	*Message*
$\neg\Phi$	*Negation*
$\Phi_1 \wedge \Phi_2$	*Conjunction*
$\Phi_1 \mid \Phi_2$	*Composition*
$\Phi_1 \triangleright \Phi_2$	*Guarantee*
$\forall x.\Phi$	*First-order Universal Quantification*
$\forall X.\Phi$	*Second-order Universal Quantification*
X	*Propositional Variable*
$\langle\alpha\rangle\Phi$	*Action*

Some derived operators used are given in the next definition.

Definition 2.2 (Some Derived Operators)

$$\begin{array}{rcl}
\Box\Phi & \triangleq & \neg\langle\tau\rangle\Phi \\
?\Phi & \triangleq & \Phi \mid \mathbf{T} \\
\bigstar\Phi & \triangleq & \neg(\neg\Phi \mid \mathbf{T}) \\
\mathbf{1} & \triangleq & \neg\mathbf{0} \wedge \neg(\neg\mathbf{0} \mid \neg\mathbf{0})
\end{array}$$

The $\Box$ operator is the dual of the diamond operator in modal temporal logic. The formula $?\Phi$ is satisfied by every process that contains some component that satisfies Φ; $\bigstar\Phi$ is satisfied by every process such that every component satisfies Φ, that is, $?$ is an existential and $\bigstar$ is an universal quantifier concerning structure; the property $\mathbf{1}$ is satisfied by all nonempty and indecomposable processes. For example, an output message in the asynchronous π-calculus satisfies this formula.

Some implications and equivalences of the spatial logic are given next.

Lemma 2.3 *For every formula Φ, Ψ and Θ, and every message m, we have*

$$\begin{array}{lcl}
\Phi \mid \mathbf{0} & \Longleftrightarrow & \Phi \\
\Phi \mid \Psi & \Longleftrightarrow & \Psi \mid \Phi \\
(\Phi \mid \Psi) \mid \Theta & \Longleftrightarrow & \Phi \mid (\Psi \mid \Theta) \\
\Phi \wedge \Psi \mid \Theta & \Longrightarrow & (\Phi \mid \Theta) \wedge (\Psi \mid \Theta) \\
\Phi \vee \Psi \mid \Theta & \Longleftrightarrow & (\Phi \mid \Theta) \vee (\Psi \mid \Theta) \\
(\Phi \mid \Psi) \wedge (\Phi \Rightarrow \Theta) & \Longrightarrow & \Theta \mid \Psi \\
[m]\Phi \wedge \langle m\rangle\Psi & \Longrightarrow & \langle m\rangle(\Phi \wedge \Psi) \\
\Phi \mid (\Phi \triangleright \Psi) & \Longrightarrow & \Psi \\
?(\Phi \mid \Psi) & \Longleftrightarrow & ?\Phi \mid ?\Psi \\
?(\Phi \wedge \mathbf{1} \mid \Psi) & \Longrightarrow & \Phi \wedge \mathbf{1} \mid ?\Psi
\end{array}$$

We now describe the syntax of processes in the asynchronous π-calculus, with summation and with possibly recursive process definitions, but without restriction and replication, which is the intended model of the spatial logic of Def. 2.1.

Definition 2.4 (The Asynchronous π-calculus)

$$P \ ::= \ \mathbf{0} \mid \overline{x}\langle\vec{y}\rangle \mid x(\vec{y}).P \mid \tau.P \mid P_1 + P_2 \mid (P_1|P_2) \mid D\langle\vec{y}\rangle$$

D is some process identifier such that the body of the definition is also a process, possibly involving other process identifiers or D itself.

Given some process of the asynchronous π-calculus we can say that it satisfies some formula of the spatial logic, according to the following table

$$\begin{array}{rcll}
\mathbf{0} & \models & \mathbf{0} & \\
\overline{x}\langle\vec{y}\rangle & \models & x\langle\vec{y}\rangle & \\
x(\vec{y}).P & \models & \forall\, \alpha.[\alpha](\alpha = x(\vec{y}) \wedge \Phi) & \text{such that } P \models \Phi \\
\tau.P & \models & \forall\, \alpha.[\alpha](\alpha = \tau \wedge \Phi) & \text{such that } P \models \Phi \\
M_1 + M_2 + \ldots + M_n & \models & \forall\, \alpha.[\alpha](\Phi_1 \vee \Phi_2 \vee \ldots \vee \Phi_n) & \text{such that } \ M_1 \models \forall\, \alpha.[\alpha]\Phi_1, \\
 & & & M_2 \models \forall\, \alpha.[\alpha]\Phi_2, \ldots \\
 & & & M_n \models \forall\, \alpha.[\alpha]\Phi_n \\
P_1 \mid P_2 & \models & \Phi_1 \mid \Phi_2 & \text{such that } P_1 \models \Phi_1 \text{ and } P_2 \models \Phi_2 \\
D\langle\vec{y}\rangle & \models & \Phi & \text{such that } D(\vec{x}) \triangleq P \\
 & & & \text{and } P\{\vec{x}\leftarrow\vec{y}\} \models \Phi
\end{array}$$

The interaction between two processes in the asynchronous π-calculus is described by the following rule $\overline{x}\langle\vec{y}\rangle \mid (x(\vec{z}).P + M) \to P\{\vec{y}/\vec{z}\}$. An internal transition is described by the rule $\tau.P + M \to P$. The next two lemmas express these transitions in the spatial logic.

Lemma 2.5 $\langle m\rangle\Phi \iff \exists\, X.\ (X \mid \mathbf{1} \wedge \langle m\rangle(X \triangleright \Phi))$

Lemma 2.6 $\langle\tau\rangle\Phi \iff \exists\, m.\ (m\langle\rangle \mid \langle m()\rangle\Phi) \ \vee\ \exists\, X.\ (X \mid \mathbf{1} \wedge \langle\tau\rangle(X \triangleright \Phi))$

3 The Arrow Distributed Directory Protocol

The Arrow Distributed Directory Protocol[7] is a peer-2-peer distributed algorithm for directory lookup. The network is composed of a set of nodes, one of which holds the object, while the others concurrently ask for its ownership. Initially, a minimum spanning tree is obtained from the whole network (Fig. 1), whose connections have costs, in such a way that the object stays at the root.

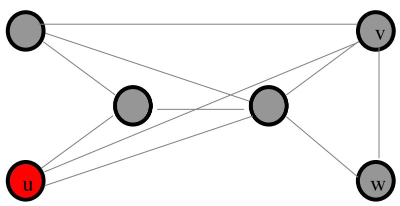

Figure 1: The whole network

A graph, called the *arrow directory*, is obtained by directing the links of this tree, in order that the path from any node leads to the one that holds the object (Fig. 2).

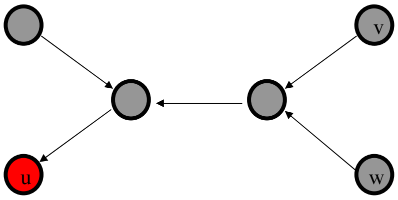

Figure 2: The arrow directory

Consequently, all the nodes, except the owner, become non-terminal and pointing to just another one.

Nodes communicate via two types of messages, the request for the object and the transfer of ownership. A request is sent by some node to the one to which it is connected to via the arrow directory, whether it's the node itself that is asking for the object or it is the propagation of some other request.

The arrow directory changes when nodes react to the requests sent to them, or by an internal transition that originates a new request which will be propagated through the graph. Every node that is not the owner or a waiter - a node that has already requested the object and hasn't received it yet - is a potential requester for the object.

When some node wants the object, it sends a message to the node to which it is connected to (Fig. 3).

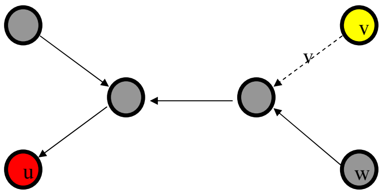

Figure 3: Node v requests the object

If some node receives a request and it is not terminal, it propagates the request to the node to which it is linked (Fig. 4).

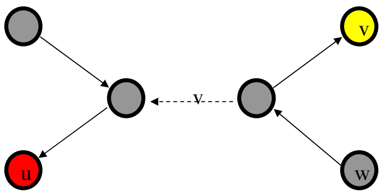

Figure 4: Propagation of the request from v

When a terminal node receives a request, it keeps it until it receives the object itself, or if it is already the owner, when it is able to send the object (Fig. 5). The transfer of ownership (Fig. 6) is done via the whole network, probably not following the same path as the request.

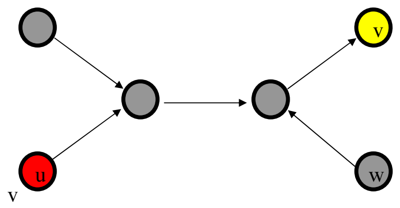

Figure 5: The request reaches terminal node u, the owner

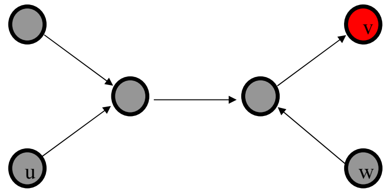

Figure 6: The object is moved to v

There can be two nodes waiting for the object at the same time. This is what happens, for example, in the configuration of Fig. 4, if node w also requests the object; however now the request is conducted to terminal node v (Fig. 7). In this way, the requests are conducted to terminal nodes, either the owner or a waiter.

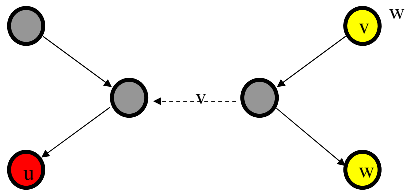

Figure 7: Concurrency in requesting the object

So, the link from a node changes either when the node asks for the object, by sending a request to the one it is connected to, while at the same time becomes terminal, or when it receives a request from another, by setting the link to that node, and by buffering the request, if it is terminal, or by propagating the request to the old link, if it is non-terminal.

At each time, there can be more than one connected component of the graph defined by the initial spanning tree (Figs. 4 and 8), but they do never become independent, as a link is always substituted by a message between the nodes, and when the receiver of the message reacts to it, it will establish the link in the opposite direction. So, in fact there aren't new established links; sometimes a link become *transparent*, and then its reappear pointing back.

Not only nodes in the same component but also in different components can concurrently ask for the object, which is what happens when, in configuration of Fig. 7, node z

also asks for the ownership (Fig. 8).

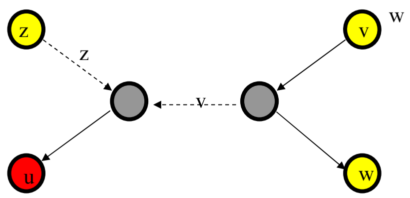

Figure 8: Another concurrent request

In the case that the request form node z is treated before the one from v, it reaches the owner and a queue of pending requests gets distributed through the network, while the requests reach terminal nodes (Fig. 9).

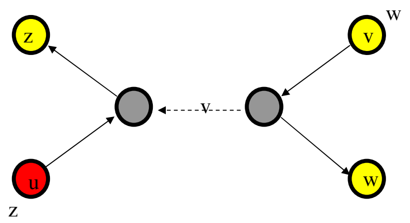

Figure 9: A queue of pending requests(I)

Note that every connected component has only one terminal node. When all pending requests are treated the arrow directory becomes as in Fig. 10. Now, we have only one component again and a terminal node w.

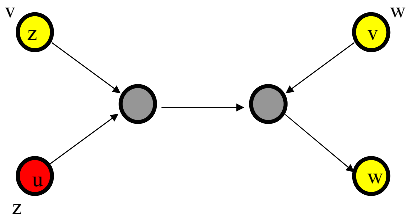

Figure 10: A queue of pending requests(II)

The transfer of ownership of the object follows the queue, first to node z, then to v and finally to w, as illustrated by the three figures (Figs. 11, 12 and 13).

Concerning ownership and pending requests, nodes can be of five types: a terminal waiter, that is, a node that has made a request and hasn't received a request from another node yet; a non-terminal waiter, which buffers a request that has already reached it; the owner, which can be a terminal node; the owner, not terminal, meaning that it has already received a request, but hasn't released the object yet; finally a potential requester, always non-terminal.

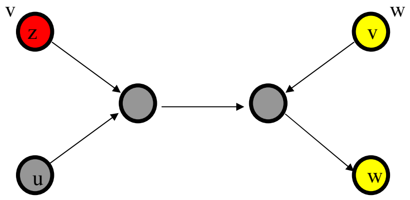

Figure 11: Transfer of ownership to z

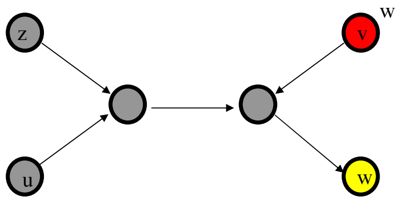

Figure 12: Transfer of ownership to v

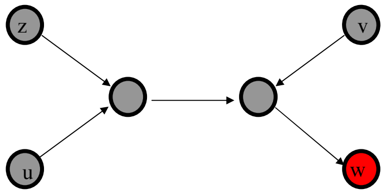

Figure 13: Transfer of ownership to w

All the types of nodes can have pending messages corresponding to requests from other nodes and are able to react accordingly. Terminal nodes buffer the request, while non-terminal nodes propagate the request to their link and become linked to the sender. Additionally, if the owner is non-terminal, it has a buffered request, so it is able to send the object. A waiter, besides its capacity to propagate requests, is able to receive the object. Finally, every node that is not the owner or a waiter is a potential requester, so at any time it can send a request to the node it is linked to.

In order to prove the correctness of the algorithm, we have to prove that every node that asks for the object, will eventually get it. Assuming that there aren't failures, that is, no message is lost and a node never fails, the path from every node always reaches a terminal node, either the owner or some node waiting for the object, so the requester will always get the object, because on the graph that constitutes the directory, the path from every node always ends on a terminal.

4 A Spatial and Behavioral Specification

In order to model the different states nodes can be on and their behavior accordingly, as described previously, we distinguish five kinds of nodes: *Owner*, *TermOwner*, *Waiter*,

1
t]

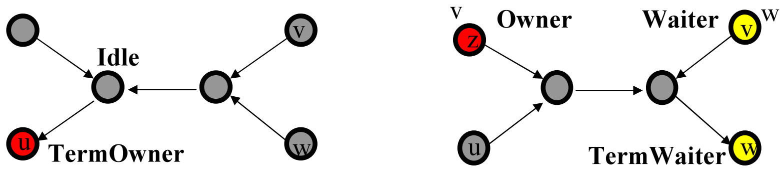

Figure 14: Node's possible states

Every node is characterized by two channels: the channel through which it receives the requests from the nodes linked to it ($find$), and the channel through which it will receive the object ($move$), when requested. Each node also keeps its link ($link$), if not terminal, which is the $find$ channel of the node to which it points to, and possibly the $move$ channel of the node that has originated a request conducted to it ($queue$). Note that after having queued a request, no other request is conducted to the same node, because it is no longer terminal unless, of course, after having received the object, transferred it to its queue and requested the object again.

In order to exemplify the description of the distinct nodes' states on the π-*calculus*, we define the process $IdleN$, representing a potential requester.

$$
\begin{aligned}
IdleN(find, move, link) \quad = \quad & find(ask, from).(IdleN(find, move, from) \mid \\
& \qquad \overline{link}\langle ask, find\rangle) + \\
& \tau.(TermWaiterN(find, move) \mid link!(move, find))
\end{aligned}
$$

In what concerns messages, a request has a target (a *find* channel) and two parameters, the *move* channel of the origin of the request and the *find* channel of the last propagator of the request, while the message representing transfer of ownership only has a target (a *move* channel).

4.1 The Node's Behavioral Specification and Some Basic Properties

The following definition is a specification for each one of the five kinds of nodes. Every equivalence says that a node is not decomposable, or it is unitary, and every possible transition of the component, through some action α, is described by the right-hand formula of the conjunction.

Definition 4.1 (Nodes Specification)

$$
\begin{array}{l}
Spec \triangleq \\
(\ TermOwner(find, move) \ \Leftrightarrow \ \mathbf{1} \ \wedge \\
\qquad \forall\ \alpha.[\alpha](\exists\ ask, from.\ \alpha = find(ask, from) \ \wedge \\
\qquad\qquad\qquad Owner(find, move, from, ask))\ ) \ \wedge \\
(\ Owner(find, move, link, queue) \ \Leftrightarrow \ \mathbf{1} \ \wedge \\
\qquad \forall\ \alpha.[\alpha]((\exists\ ask, from.\ \alpha = find(ask, from) \ \wedge \\
\qquad\qquad\qquad Owner(find, move, from, queue) \mid link\langle ask, find\rangle) \vee \\
\qquad\qquad (\alpha = \tau \ \wedge \ Idle(find, move, link) \mid queue\langle\rangle))\ ) \ \wedge \\
(\ Idle(find, move, link) \ \Leftrightarrow \ \mathbf{1} \ \wedge \\
\qquad \forall\ \alpha.[\alpha]((\exists\ ask, from.\ \alpha = find(ask, from) \ \wedge \\
\qquad\qquad\qquad Idle(find, move, from) \mid link\langle ask, find\rangle) \vee \\
\qquad\qquad (\alpha = \tau \ \wedge \ TermWaiter(find, move) \mid link\langle move, find\rangle))\ ) \ \wedge \\
(\ TermWaiter(find, move) \ \Leftrightarrow \ \mathbf{1} \ \wedge \\
\qquad \forall\ \alpha.[\alpha]((\exists\ ask, from.\ \alpha = find(ask, from) \ \wedge \\
\qquad\qquad\qquad Waiter(find, move, from, ask)) \vee \\
\qquad\qquad (\alpha = move() \ \wedge \ TermOwner(find, move)))\ ) \ \wedge \\
(\ Waiter(find, move, link, queue) \ \Leftrightarrow \ \mathbf{1} \ \wedge \\
\qquad \forall\ \alpha.[\alpha]((\exists\ ask, from.\ \alpha = find(ask, from) \ \wedge \\
\qquad\qquad\qquad Waiter(find, move, from, queue) \mid link\langle ask, find\rangle) \vee \\
\qquad\qquad (\alpha = move() \ \wedge \ Owner(find, move, link, queue)))\ )
\end{array}
$$

Every node is able to react to a message representing a request, by the input action $find\langle ask, from\rangle$, and then it becomes linked to the node that sent the request, while it propagates the request, if not terminal; $Idle$ nodes are also able to request the object, through an internal τ-transition; $Owner$ nodes, which have a pending request, can send the object to the node identified by the channel $queue$; $Waiter$ and $TermWaiter$ nodes are able to receive the object through their $move$ channel, by the input action $move()$.

Other definitions concerning nodes and messages follow.

Definition 4.2 (Nodes and Messages)

$$
\begin{array}{rcl}
TerminalNode(find) & \Leftrightarrow & \exists\ move.\ TermOwner(find, move) \ \vee \\
 & & \exists\ move.\ TermWaiter(find, move) \\
LinkedNode(find, link) & \Leftrightarrow & \exists\ move.\ Idle(find, move, link) \ \vee \\
 & & \exists\ move, queue.\ Owner(find, move, link, queue) \ \vee \\
 & & \exists\ move, queue.\ Waiter(find, move, link, queue) \\
Node(find) & \Leftrightarrow & TerminalNode(find) \ \vee \ \exists\ link.\ LinkedNode(find, link) \\
Node & \Leftrightarrow & \exists\ find.\ Node(find) \\
Msg & \Leftrightarrow & \exists\ move.\ move\langle\rangle \ \vee \ \exists\ find, move, link.\ find\langle move, link\rangle
\end{array}
$$

4.2 Graph-Related Properties

We have already defined what kind of nodes can be found in the graph that constitutes the network and how they behave. In this section we present some definitions concerning the links between them. We say that two nodes are linked if there is a *physical* connection or a message in transit between them; the notion of undirected path is defined recursively, by making use of the spatial composition operator, which allows us to express a path as being a system with two components, a link and some other path; finally, we define what a physical path is, where the links between the nodes are only established by their connections and not also by the pending requests between them.

Definition 4.3 (Graphs)

$$\begin{array}{rcl} Link(find_1, find_2) & \triangleq & LinkedNode(find_1, find_2) \ \vee \ \exists\, find_3.\ find_2\langle find_3, find_1\rangle \\ UndirPath(find_1, find_2) & \triangleq & find_1 = find_2 \ \vee \ \exists\, find_3.\ (Link(find_1, find_3) \ \vee \\ & & Link(find_3, find_1)) \mid UndirPath(find_3, find_2) \\ PhyPath(find_1, find_2) & \triangleq & find_1 = find_2 \ \vee \\ & & \exists\, find_3.\ LinkedNode(find_1, find_3) \mid PhyPath(find_3, find_2) \end{array}$$

The next property states that there are never two alternative undirected paths between any two nodes. The use of the operator **?** to qualify a property means that it can possibly be satisfied by just some part of the system.

Definition 4.4 (Unique Undirected Path)

$$\begin{array}{rcl} UniqueUndirPath & \triangleq & \neg\, \exists\, find_1, find_2.\ find_1 \neq find_2 \ \wedge \ (?UndirPath(find_1, find_2) \mid \\ & & \hfill ?UndirPath(find_1, find_2)) \end{array}$$

Note that, as a consequence of this property, we can not only deduce that, if there exists a path between two nodes, it is unique, but also that there are no cycles on the directed graph, because every directed path is, by definition, an undirect path too, where by a directed path we mean a chain of connections between the nodes.

4.3 The Initial State and the Safety Properties

The initial state of the system, that is, the graph determined by the minimal spanning tree obtained in the beginning, must always satisfy the following formula. The specification is not compromised with any particular configuration, concerning the number of nodes and the links between them, but only with the existence of just one node owning the object and all the others being potential requesters.

Definition 4.5 (Initial State)

$$\begin{array}{rcl} Init & \triangleq & \exists\, findOw, moveOw.\ TermOwner(findOw, moveOw) \mid \\ & & \bigstar(\mathbf{1} \ \Rightarrow \ \exists\, find, move, link.\ Idle(find, move, link)) \ \wedge \\ & & (\forall\, node.\ ?Node(node) \ \Rightarrow \ ?PhyPath(node, findOw)) \end{array}$$

The last condition guarantees that the initial configuration is indeed a spanning tree, because every node is only linked to another one and the implication expresses that it has a path to the owner.

Not only the nodes, but also the messages, compose the system while it evolves, and both constitute the unitary components that can interact with each other, according to the properties stated in Lemmas 2.5 and 2.6.

The property *UnitComp* states that every unitary component is either a node or a message. It will be proved in the following section that this property is an invariant of a system that satisfies *Init* and where *Spec*, the nodes's specification, is also an invariant.

Definition 4.6 (Atomic Components)

$$UnitComp \ \triangleq \ \bigstar(\mathbf{1} \Rightarrow Node \vee Msg)$$

The next definition is well-defined for every node, because *UniqueUndirPath* is an invariant, as we also prove in the next section, which means that there is always just one undirected path between any two nodes. In consequence, there are no cycles on the graph constituted by the nodes and their connections, and the path starting from any node is unique.

Definition 4.7 (Path from a Node)

$$Target(x) \triangleq TerminalNode(x) \ \vee \ (\exists\, y.\ LinkedNode(x,y) \mid Target(y))$$

Note that $Target(x)$ is true only for the part of the system composed by the nodes that constitute the path leading to a terminal node. We prove that for every node x, there is always some component of the system that satisfies the property $Target(x)$.

The next property is the safety property that we want to establish at the end, because it ensures that any request will always be conducted to a terminal node, which can be the owner($TermOwner$) or a waiter($TermWaiter$), by definition.

Definition 4.8 (Safety Property)

$$Safety \triangleq \forall\, node.\ ?Node(node) \ \Rightarrow \ ?Target(node)$$

4.4 Some Lemmas Related to the $UnitComp$ and $Safety$ Properties

The next two lemmas are used to prove that $UnitComp$ is an invariant for a system with an initial state satisfying $Init$ and with the nodes's specification $Spec$.

The following lemma states the preservation of the property $UnitComp$, when some unitary component is added to the system.

Lemma 4.9 $UnitComp \mid Node \vee Msg \ \Rightarrow \ UnitComp$

Proof. Assume that $UnitComp \mid Node \vee Msg$ and also $\neg UnitComp$. The last formula is $\neg\bigstar(\mathbf{1} \Rightarrow Node \vee Msg)$, by the definition of $UnitComp$, that is, $\neg(\mathbf{1} \Rightarrow Node \vee Msg) \mid \mathbf{T}$, which implies $(\mathbf{1} \wedge \neg Node \wedge \neg Msg) \mid \mathbf{T}$. By the antecedent, we obtain

$$\neg(\neg(\mathbf{1} \Rightarrow Node \vee Msg) \mid \mathbf{T}) \mid Node \vee Msg$$

which implies

$$\neg((\mathbf{1} \wedge \neg Node \wedge \neg Msg) \mid \mathbf{T}) \mid Node \vee Msg$$

that is

$$\neg((\mathbf{1} \wedge \neg Node \wedge \neg Msg) \mid \mathbf{T}) \mid \mathbf{1} \ \wedge \ (Node \vee Msg)$$

and then $\neg((\mathbf{1} \wedge \neg Node \wedge \neg Msg) \mid \mathbf{T})$. So, the antecedent in conjunction with $\neg UnitComp$ leads to a contradiction. $\Box$

In the following lemma, we infer the local preservation of $UnitComp$ when it is true globally.

Lemma 4.10 $UnitComp \ \Rightarrow \ \bigstar UnitComp$

Proof. Assume that $UnitComp$ and $\neg\bigstar UnitComp$, which implies

$$UnitComp \ \wedge \ (\neg UnitComp \mid \mathbf{T})$$

that is, $\neg((\mathbf{1} \wedge \neg Node \wedge \neg Msg) \mid \mathbf{T}) \ \wedge \ ((\mathbf{1} \wedge \neg Node \wedge \neg Msg) \mid \mathbf{T})$, which implies $\mathbf{F}$. $\Box$

We present next an equivalent definition for $Target(x)$.

Proof. Let's prove the equivalence from left to right. Assume that $Target(x)$. Then, by the definition of $Target(x)$

$$TerminalNode(x) \ \vee \ (\exists\, y.\ LinkedNode(x,y) \mid Target(y))$$

If $TerminalNode(x)$ then $PhyPath(x, x) \wedge ?TerminalNode(x)$. Otherwise, by the induction hypothesis, we get $\exists\, z.\ PhyPath(y, z) \ \wedge\ ?TerminalNode(z)$. Now, by the definition of $PhyPath$ we obtain $?PhyPath(x, z) \ \wedge\ ?TerminalNode(z)$, and the result follows. Now, let's prove the implication on the other way. Assume that

$$\exists\, y.\ PhyPath(x, y) \ \wedge\ ?TerminalNode(y)$$

By the definition of $PhyPath$, either $x = y$ or $\exists\, z.\ LinkedNode(x, z) \mid PhyPath(z, y)$. If $x = y$ then $TerminalNode(x)$ and so $Target(x)$. Otherwise, by the induction hypothesis, $Target(z)$, and then, by the definition, we obtain $Target(x)$. □

If some node belongs to the path that starts on one node, then there must be a path between the later and the former, appended to a path leading to a terminal. The next lemma states this property.

Lemma 4.11

$$\begin{aligned}\forall\, x, y.\ ?Node(x) \ \wedge\ Target(y) \ \Rightarrow\ (&PhyPath(y, x) \mid \\ &\exists\, z.\ PhyPath(x, z) \ \wedge\ ?TerminalNode(z))\end{aligned}$$

Proof. If $x = y$ the antecedent becomes $?Node(y) \ \wedge\ Target(y)$ and then, by definition, $TerminalNode(y)$ and the consequent follows by choosing $z = x = y$. Otherwise, $\exists\, w.\ LinkedNode(y, w) \mid Target(w)$. Now, if $w = x$ we get $PhyPath(y, x) \mid Target(x)$, and then if $?TerminalNode(x)$, it follows that

$$PhyPath(y, x) \mid PhyPath(x, x) \ \wedge\ ?TerminalNode(x)$$

or else, by the induction hypothesis, it follows that

$$LinkedNode(y, w) \mid PhyPath(w, x) \mid \exists\, z.\ PhyPath(x, z) \ \wedge\ ?TerminalNode(z)$$

and then

$$PhyPath(y, x) \mid \exists\, z.\ PhyPath(x, z) \ \wedge\ ?TerminalNode(z)$$

by the definition of $PhyPath$, and the result follows. □

4.5 Some Statements about Graphs

In this section we present some auxiliary statements about undirected graphs, where the vertices are the nodes and the edges are the connections, as well as the requests in transit between the nodes. These lemmas are used to prove that $UniqueUndirPath$ is an invariant of the system.

The next lemma states that a terminal node doesn't add any undirected path.

Lemma 4.12

$$\begin{aligned}\forall\, n, x, y.\ ?TerminalNode(n) \ \wedge\ ?UndirPath(x, y) \ \Rightarrow& \\ TerminalNode(n) \mid ?UndirPath(x, y)&\end{aligned}$$

Proof. The proof is done by induction on the definition of $UndirPath(x, y)$. If $x = y$, the result follows because $?UndirPath(x, x)$ is true everywhere, by definition. If $x \neq y$ then $\exists\, z.\ (Link(x, z) \ \vee\ Link(z, x) \mid ?UndirPath(z, y))$. So, the antecedent becomes

$$?TerminalNode(n) \ \wedge\ \exists\, z.\ (Link(x, z) \ \vee\ Link(z, x) \mid ?UndirPath(z, y))$$

By the induction hypothesis we obtain

$$\exists\, z.\ \ TerminalNode(n) \mid ?UndirPath(z, y) \mid Link(x, z)\ \vee\ Link(z, x)$$

noting that the components $TerminalNode(n)$ and $Link(x, z) \vee Link(z, x)$ must be distinct, because n is a node and terminal, and so the consequent follows from the definition of $UndirPath(x, y)$. □

The following lemma states that if there exist an undirected path between nodes x and y and another one between a and b and, then either one belongs to the other or not.

Lemma 4.13

$$\begin{aligned} \forall\, a, b, x, y.\ \ ?UndirPath(x, y)\ \wedge\ ?UndirPath(a, b)\ \Rightarrow & \\ & (?UndirPath(x, a) \mid UndirPath(a, b) \mid ?UndirPath(b, y))\ \vee \\ & (?UndirPath(x, b) \mid UndirPath(a, b) \mid ?UndirPath(a, y))\ \vee \\ & (?UndirPath(x, y) \mid UndirPath(a, b)) \end{aligned}$$

Proof. The proof is done by induction on the definition of $UndirPath(x, y)$. If $x = y$ the result follows, because $?UndirPath(x, x)$ is true everywhere, and so

$$?UndirPath(x, x)\ \wedge\ ?UndirPath(a, b)\ \Rightarrow\ ?UndirPath(x, x) \mid UndirPath(a, b)$$

If $x \neq y$, the antecedent implies

$$?UndirPath(a, b)\ \wedge\ \exists\, z.\ ((Link(x, z)\ \vee\ Link(z, x)) \mid ?UndirPath(z, y))$$

which expands in the two cases

$$?UndirPath(a, b)\ \wedge\ \exists\, z.\ (Link(x, z) \mid ?UndirPath(z, y)) \tag{1}$$

and

$$?UndirPath(a, b)\ \wedge\ \exists\, z.\ (Link(z, x) \mid ?UndirPath(z, y)) \tag{2}$$

For case (1), if $x = a$ and $z = b$ then the antecedent implies

$$?UndirPath(x, a) \mid UndirPath(a, b) \mid ?UndirPath(b, y)$$

Otherwise, if $x = b$ and $z = a$ then the formula

$$?UndirPath(x, b) \mid UndirPath(a, b) \mid ?UndirPath(a, y)$$

follows. Else, by applying the induction hypothesis, we obtain

$$\begin{aligned} & (?UndirPath(z, a) \mid UndirPath(a, b) \mid ?UndirPath(b, y) \mid Link(x, z))\ \vee \\ & (?UndirPath(z, b) \mid UndirPath(a, b) \mid ?UndirPath(a, y) \mid Link(x, z))\ \vee \\ & (?UndirPath(z, y) \mid UndirPath(a, b) \mid Link(x, z)) \end{aligned}$$

Now, by the definition of $UndirPath$, the first formula of the disjunction implies

$$?UndirPath(x, a) \mid UndirPath(a, b) \mid ?UndirPath(b, y)$$

the second formula implies

$$?UndirPath(x, b) \mid UndirPath(a, b) \mid ?UndirPath(a, y)$$

and the third one implies $?UndirPath(x, y) \mid UndirPath(a, b)$, and the result follows.

Now, concerning case (2), the result follows by a similar reasoning. □

The next lemma states the transitivity of the property $UndirPath$.

Lemma 4.14

$$\forall\ x, y, z.\ \ ?UndirPath(x, y)\ \wedge\ ?UndirPath(y, z)\ \Rightarrow\ ?UndirPath(x, z)$$

Proof. We prove this lemma by induction on the definition of $UndirPath(x, y)$. If $x = y$ the result follows directly. Otherwise

$$(\exists\ w.\ (Link(x, w)\ \vee\ Link(w, x))\ |\ ?UndirPath(w, y))\ \wedge\ ?UndirPath(y, z)$$

and then, by applying the induction hypothesis, we obtain

$$\exists\ w.\ (Link(x, w)\ \vee\ Link(w, x))\ |\ ?UndirPath(w, z)$$

and then $?UndirPath(x, z)$, by the definition of $UndirPath$. $\square$

The property $UndirPath$ is also symmetric, as it is proved in the lemma that follows.

Lemma 4.15

$$\forall\ x, y.\ \ UndirPath(x, y)\ \Rightarrow\ UndirPath(y, x)$$

Proof. If $x = y$ the result follows immediately. Else

$$\exists\ z.\ (Link(x, z)\ \vee\ Link(z, x))\ |\ UndirPath(z, y)$$

By the induction hypothesis, we obtain

$$\exists\ z.\ (Link(x, z)\ \vee\ Link(z, x))\ |\ UndirPath(y, z)$$

Now, by the definition of $UndirPath$ we obtain

$$UndirPath(z, x)\ |\ UndirPath(y, z)$$

Then, applying Lemma 4.14, we obtain $UndirPath(y, x)$. $\square$

The following lemma states the preservation of the property $UniqueUndirPath$ when some terminal node is added to the system.

Lemma 4.16

$$\forall\ find.\ \ UniqueUndirPath\ |\ TerminalNode(find)\ \Rightarrow\ UniqueUndirPath$$

Proof. Assume by hypothesis that the antecedent is true, and also that $\neg UniqueUndirPath$, that is,

$$\exists\ f_1, f_2.\ \ f_1 \neq f_2\ \wedge\ (?UndirPath(f_1, f_2)\ |\ ?UndirPath(f_1, f_2))$$

By hypothesis and by applying Lemma 4.12 twice to the previous formula, we obtain

$$?UndirPath(f_1, f_2)\ |\ ?UndirPath(f_1, f_2)\ |\ TerminalNode(find)$$

and then $\neg UniqueUndirPath\ |\ TerminalNode(find)$, that contradicts the antecedent. $\square$

The following lemma also guarantees the preservation of $UniqueUndirPath$, but now when a component with some undirected path between two nodes is added.

Lemma 4.17

$$\forall\ find, link.\ \ (UniqueUndirPath\ \wedge\ \neg ?UndirPath(find, link)\ |\\ UndirPath(find, link))\ \Rightarrow\ UniqueUndirPath$$

Proof. Assume, by hypothesis, that the antecedent and $\neg UniqueUndirPath$ are true. Then

$$\exists\, x, y.\ \ x \neq y\ \wedge\ ?UndirPath(x, y) \mid ?UndirPath(x, y)$$

The case of $x = find$ and $y = link$ is impossible by $\neg ?UndirPath(find, link)$ in the antecedent. Now, the formula $?UndirPath(x, y)\ \wedge\ ?UndirPath(find, link)$, which is true by hypothesis, implies

$$\begin{array}{l}(?UndirPath(x, find) \mid UndirPath(find, link) \mid ?UndirPath(link, y) \mid ?UndirPath(x, y))\ \vee\\ (?UndirPath(x, link) \mid UndirPath(find, link) \mid ?UndirPath(find, y) \mid ?UndirPath(x, y))\ \vee\\ (?UndirPath(x, y) \mid UndirPath(find, link) \mid ?UndirPath(x, y))\end{array}$$

by the application of Lemma 4.13. Then, the third case implies

$$UndirPath(find, link) \mid \neg UniqueUndirPath$$

which contradicts the antecedent. Concerning the first and second cases, applying Lemmas 4.14 and 4.15, we obtain $?UndirPath(link, find) \mid UndirPath(find, link)$ contradicting

$$\neg ?UndirPath(find, link) \mid UndirPath(find, link)$$

in the antecedent. $\Box$

In the following lemma, we state that if the property $UniqueUndirPath$ is true globally and there is an undirected path between two nodes in some part of the system, then the other part can not have a path between the same pair of nodes.

Lemma 4.18

$$\begin{array}{l}\forall\ find, link.\ UniqueUndirPath\ \wedge\ ?UndirPath(find, link)\ \Rightarrow\\ \qquad\qquad \neg ?UndirPath(find, link) \mid UndirPath(find, link)\end{array}$$

Proof. Assume $UniqueUndirPath\ \wedge\ ?UndirPath(find, link)$. Now, assume that

$$?UndirPath(find, link) \mid UndirPath(find, link)$$

Obviously, it contradicts the hypothesis that $UniqueUndirPath$ is true, so we conclude that $\neg(?UndirPath(find, link) \mid UndirPath(find, link))$, and the consequent follows. $\Box$

The following lemma states the preservation of the property $UniqueUndirPath$ on every part of the system, if it is true globally.

Lemma 4.19

$$UniqueUndirPath\ \Rightarrow\ \bigstar UniqueUndirPath$$

Proof. Assume that $UniqueUndirPath$ and $\neg\bigstar UniqueUndirPath$. The second formula implies $\neg UniqueUndirPath \mid \mathbf{T}$, by the definition of the $\bigstar$ operator. Then, it follows that

$$\exists\, x, y.\ x \neq y\ \wedge\ ?UndirPath(x, y)\ \wedge\ ?UndirPath(x, y) \mid \mathbf{T}$$

that is,

$$\exists\, x, y.\ x \neq y\ \wedge\ ?UndirPath(x, y)\ \wedge\ ?UndirPath(y, x)$$

or $\neg UniqueUndirPath$, that contradicts the antecedent. $\Box$

5 A Proof for *Safety* Properties

Suppose we have a formula S describing the unitary components of the system and we want to prove the invariant Φ for a system where the initial state satisfies the formula I, that is, we want to prove that $I \wedge S \Rightarrow \Box^*\Phi$. $\Box^*$ is the operator we use to express that some property is an invariant. The following rule allows us to establish the desired result.

$$\frac{I \wedge S \Rightarrow \Phi \quad S \wedge \Phi \Rightarrow \Box\Phi}{I \wedge S \Rightarrow \Box^*\Phi}$$

The previous rule expresses that some property is true on every state, if it is true in the initial state, and also if assuming that it is true on some state it is also satisfied on every one-step transition state reached from it.

In this section we prove that $UnitComp$, $UniqueUndirPath$ and $Safety$ are safety properties for a system with some initial structure that satisfies $Init$ and such that the behavior of its components is specified by the invariant $Spec$.

5.1 Proof of $UnitComp$

We prove that $UnitComp$ is an invariant, by using the previous rule, as well as by applying Lemmas 4.9 and 4.10. First, we prove that $Init \Rightarrow UnitComp$, as we don't need $Spec$ in the antecedent to derive $UnitComp$.

Lemma 5.1 $Init \;\Rightarrow\; UnitComp$

Proof. We have that

$$\begin{array}{lll} Init & \Rightarrow & \mathbf{1} \wedge (\exists\, findOw, moveOw.\ TermOwner(findOw, moveOw)) \mid \\ & & \bigstar(\mathbf{1} \;\Rightarrow\; \exists\, find, move, link.\ Idle(find, move, link)) \\ & \Rightarrow & \bigstar(\mathbf{1} \;\Rightarrow\; Node) \end{array}$$

which then implies $UnitComp$. $\Box$

The next lemma proves that every state that satisfies $UnitComp$ can only transit to another state where the property is preserved, according to the nodes' specification $Spec$.

Lemma 5.2 $Spec \wedge UnitComp \;\Rightarrow\; \Box UnitComp$

Proof. Let's assume that $Spec \;\wedge\; UnitComp \;\wedge\; \langle\rangle\neg UnitComp$. The previous formula implies

$$\begin{array}{r} Spec \;\wedge\; UnitComp \;\wedge\; (\exists\, m.\ (m\langle\rangle \mid \langle m()\rangle\neg UnitComp) \;\vee \\ \exists\, X.\ (X \mid \mathbf{1} \wedge \langle\tau\rangle(X \triangleright \neg UnitComp))) \end{array}$$

by Axiom 2.6, which then implies the two cases

$$Spec \;\wedge\; UnitComp \;\wedge\; \exists\, m.\ (m\langle\rangle \mid \langle m()\rangle\neg UnitComp) \tag{3}$$

and

$$Spec \;\wedge\; UnitComp \;\wedge\; \exists\, X.\ (X \mid \mathbf{1} \wedge \langle\tau\rangle(X \triangleright \neg UnitComp)) \tag{4}$$

Now, formula (3) implies

$$Spec \;\wedge\; UnitComp \;\wedge\; \exists\, m.\ (m\langle\rangle \wedge Msg \mid \exists\, X.\ (X \mid \mathbf{1} \wedge \langle m()\rangle(X \triangleright \neg UnitComp)))$$

by $UnitComp$ and by the Axiom 2.5, and then

$$Spec \;\wedge\; UnitComp \;\wedge\; \exists\, m.\ (m\langle\rangle \wedge Msg \mid \exists\, X.\ (X \mid Node \;\wedge\; \langle m()\rangle(X \triangleright \neg UnitComp)))$$

also by $UnitComp$. Then, by Msg, the two cases follow

$$Spec \ \wedge \ UnitComp \ \wedge \ \exists \, obj. \ (obj\langle\rangle \mid \exists \, X. \ (X \mid Node \wedge \langle obj()\rangle(X \triangleright \neg UnitComp))) \quad (5)$$

and

$$\begin{aligned} &Spec \ \wedge \ UnitComp \ \wedge \ \exists \, find, move, from. \ (find\langle move, from\rangle \mid \\ &\qquad \exists \, X. \ (X \mid Node \ \wedge \ \langle find(move, from)\rangle(X \triangleright \neg UnitComp))) \end{aligned} \quad (6)$$

In the case that formula (5) is true, we obtain

$$\begin{aligned} Spec \ \wedge \ UnitComp \ \wedge \ \exists \, X. \, \exists \, obj. \ (X \mid obj\langle\rangle \mid \\ (\exists \, find, move. \ TermOwner(find, move) \ \vee \\ \exists \, find, move. \ TermWaiter(find, move) \ \vee \\ \exists \, find, move, link. \ Idle(find, move, link) \ \vee \\ \exists \, find, move, link, queue. \ Waiter(find, move, link, queue) \ \vee \\ \exists \, find, move, link, queue. \ Owner(find, move, link, queue)) \ \wedge \\ \langle obj()\rangle(X \triangleright \neg UnitComp)) \end{aligned}$$

by $Node$. Eliminating the existential quantifiers, the previous formula implies the five cases

$$\begin{aligned} &Spec \ \wedge \ UnitComp \ \wedge \ (X \mid obj\langle\rangle \mid TermOwner(find, move) \ \wedge \\ &\qquad \langle obj()\rangle(X \triangleright \neg UnitComp)) \end{aligned} \quad (7)$$

$$\begin{aligned} &Spec \ \wedge \ UnitComp \ \wedge \ (X \mid obj\langle\rangle \mid TermWaiter(find, move) \ \wedge \\ &\qquad \langle obj()\rangle(X \triangleright \neg UnitComp)) \end{aligned} \quad (8)$$

$$\begin{aligned} &Spec \ \wedge \ UnitComp \ \wedge \ (X \mid obj\langle\rangle \mid Idle(find, move, link) \ \wedge \\ &\qquad \langle obj()\rangle(X \triangleright \neg UnitComp)) \end{aligned} \quad (9)$$

$$\begin{aligned} &Spec \ \wedge \ UnitComp \ \wedge \ (X \mid obj\langle\rangle \mid Waiter(find, move, link, queue) \ \wedge \\ &\qquad \langle obj()\rangle(X \triangleright \neg UnitComp)) \end{aligned} \quad (10)$$

$$\begin{aligned} &Spec \ \wedge \ UnitComp \ \wedge \ (X \mid obj\langle\rangle \mid Owner(find, move, link, queue) \ \wedge \\ &\qquad \langle obj()\rangle(X \triangleright \neg UnitComp)) \end{aligned} \quad (11)$$

The formulas (7), (9) and (11) imply **F**, as it is illustrated next, for the case of formula (9)

$$\begin{aligned} (9) \ \Rightarrow \quad & X \mid obj\langle\rangle \mid \forall \, \alpha.[\alpha]((\exists \, ask, from. \ \alpha = find(ask, from) \wedge \\ & (Idle(find, move, from) \mid link\langle ask, find\rangle)) \ \vee \\ & (\alpha = \tau \wedge (TermWaiter(find, move) \mid link\langle ask, find\rangle))) \ \wedge \\ & \langle obj()\rangle(X \triangleright \neg UnitComp) \end{aligned}$$

Now, the previous formula implies

$$\begin{aligned} X \mid obj\langle\rangle \mid \langle obj()\rangle((\exists \, ask, from. \ obj() = find(ask, from) \wedge \\ (Idle(find, move, from) \mid link\langle ask, find\rangle)) \ \vee \\ (obj() = \tau \wedge (TermWaiter(find, move) \mid link\langle ask, find\rangle))) \ \wedge \\ (X \triangleright \neg UnitComp) \end{aligned}$$

which implies $X \mid obj\langle\rangle \mid \langle obj()\rangle\mathbf{F}$, that is $\mathbf{F}$.

Lemma 5.3 proves that also formula (8) implies $\mathbf{F}$. The case of formula (10) implying $\mathbf{F}$ is similar.

Now, formula (6) implies

$$\begin{aligned} Spec \wedge UnitComp \wedge \exists X. \exists\, find, ask, from.\ (X \mid find\langle ask, from\rangle \mid \\ (\exists\, find_1, move.\ TermOwner(find_1, move) \vee \\ \exists\, find_1, move.\ TermWaiter(find_1, move) \vee \\ \exists\, find_1, move, link.\ Idle(find_1, move, link) \vee \\ \exists\, find_1, move, link, queue.\ Waiter(find_1, move, link, queue) \vee \\ \exists\, find_1, move, link, queue.\ Owner(find_1, move, link, queue)) \wedge \\ \langle find(ask, from)\rangle(X \rhd \neg UnitComp)) \end{aligned}$$

by $Node$. Consequently, we obtain five cases

$$\begin{aligned} Spec \wedge UnitComp \wedge (X \mid find\langle ask, from\rangle \mid TermOwner(find_1, move) \wedge \\ \langle find(ask, from)\rangle(X \rhd \neg UnitComp)) \end{aligned} \tag{12}$$

$$\begin{aligned} Spec \wedge UnitComp \wedge (X \mid find\langle ask, from\rangle \mid TermWaiter(find_1, move) \wedge \\ \langle find(ask, from)\rangle(X \rhd \neg UnitComp)) \end{aligned} \tag{13}$$

$$\begin{aligned} Spec \wedge UnitComp \wedge (X \mid find\langle ask, from\rangle \mid Idle(find_1, move, link) \wedge \\ \langle find(ask, from)\rangle(X \rhd \neg UnitComp)) \end{aligned} \tag{14}$$

$$\begin{aligned} Spec \wedge UnitComp \wedge (X \mid find\langle ask, from\rangle \mid Waiter(find_1, move, link, queue) \wedge \\ \langle find(ask, from)\rangle(X \rhd \neg UnitComp)) \end{aligned} \tag{15}$$

$$\begin{aligned} Spec \wedge UnitComp \wedge (X \mid find\langle ask, from\rangle \mid Owner(find_1, move, link, queue) \wedge \\ \langle find(ask, from)\rangle(X \rhd \neg UnitComp)) \end{aligned} \tag{16}$$

Formula (12) implies $\mathbf{F}$, as proved in Lemma 5.6. The proof that formula (14) also gives a contradiction is given in Lemma 5.7. The proof of a contradiction for the other cases follow similarly to the previous cases, depending on the node being terminal or nonterminal, respectively.

Finally, the case of an internal transition on some node, when formula (4) is satisfied, will be considered. This formula implies the five next cases, as usual

$$Spec \wedge UnitComp \wedge (X \mid TermOwner(find, move) \wedge \langle\tau\rangle(X \rhd \neg UnitComp)) \tag{17}$$

$$Spec \wedge UnitComp \wedge (X \mid TermWaiter(find, move) \wedge \langle\tau\rangle(X \rhd \neg UnitComp)) \tag{18}$$

$$Spec \wedge UnitComp \wedge (X \mid Idle(find, move, link) \wedge \langle\tau\rangle(X \rhd \neg UnitComp)) \tag{19}$$

$$\begin{aligned} Spec \wedge UnitComp \wedge (X \mid Waiter(find, move, link, queue) \wedge \\ \langle\tau\rangle(X \rhd \neg UnitComp)) \end{aligned} \tag{20}$$

$$\begin{aligned} Spec \wedge UnitComp \wedge (X \mid Owner(find, move, link, queue) \wedge \\ \langle\tau\rangle(X \rhd \neg UnitComp)) \end{aligned} \tag{21}$$

The formulas (17), (18) and (20) imply $\mathbf{F}$, which can be proved similarly as we did for the case when formula (9) was true. The case that formula (19) implies $\mathbf{F}$ is proved in Lemma 5.4, and the case of formula (21) is proved in Lemma 5.5. $\square$

5.2 Auxiliary Lemmas for *UnitComp*

Lemma 5.3

$$\forall\, X.\ \forall\, v,m,n.\ \mathit{Spec}\ \wedge\ \mathit{UnitComp}\ \wedge\ (X \mid m\langle\rangle \mid \mathit{TermWaiter}(v,n)\ \wedge\ \langle m()\rangle(X \rhd \neg\mathit{UnitComp}))\ \Rightarrow\ \mathbf{F}$$

Proof. The formula in the antecedent implies

$$\mathit{Spec}\ \wedge\ (\mathit{UnitComp}\ \wedge\ X \mid m\langle\rangle \mid \mathit{TermWaiter}(v,n)\ \wedge\ \langle m()\rangle(X \rhd \neg\mathit{UnitComp}))$$

by Lemma 4.10, and then

$$\mathit{UnitComp}\ \wedge\ X \mid m\langle\rangle \mid \forall\, \alpha.[\alpha]((\exists\, w,u.\ \alpha = v(w,u)\ \wedge\ \mathit{Waiter}(v,n,u,w))\ \vee\ (\alpha = n()\ \wedge\ \mathit{TermOwner}(v,n)))\ \wedge\ \langle m()\rangle(X \rhd \neg\mathit{UnitComp})$$

by Def. 4.1, which reduces to

$$\mathit{UnitComp}\ \wedge\ X \mid m\langle\rangle \mid \langle m()\rangle((\exists\, w,u.\ m() = v(w,u)\ \wedge\ \mathit{Waiter}(v,n,u,w))\ \vee\ (m() = n()\ \wedge\ \mathit{TermOwner}(v,n)))\ \wedge\ (X \rhd \neg\mathit{UnitComp})$$

Now, if $m = n$, the formula reduces to

$$\mathit{UnitComp}\ \wedge\ X \mid \langle\tau\rangle(\mathit{TermOwner}(v,n)\ \wedge\ (X \rhd \neg\mathit{UnitComp}))$$

by Axiom 2.6, and then

$$\langle\tau\rangle((\mathit{UnitComp} \mid \mathit{TermOwner}(v,n))\ \wedge\ \neg\mathit{UnitComp})$$

which implies $\langle\tau\rangle(\mathit{UnitComp} \wedge \neg\mathit{UnitComp})$ by lemma 4.9, that is, $\langle\tau\rangle\mathbf{F}$, or $\mathbf{F}$.

If $m \neq n$ then the formula implies $\langle\tau\rangle\mathbf{F}$, that is $\mathbf{F}$. $\square$

Lemma 5.4

$$\forall\, X.\ \forall\, v,n,l.\ \mathit{Spec}\ \wedge\ \mathit{UnitComp}\ \wedge\ (X \mid \mathit{Idle}(v,n,l)\ \wedge\ \langle\tau\rangle(X \rhd \neg\mathit{UnitComp}))\ \Rightarrow\ \mathbf{F}$$

Proof. The formula in the antecedent implies

$$\mathit{Spec}\ \wedge\ (\mathit{UnitComp}\ \wedge\ X \mid \mathit{Idle}(v,n,l)\ \wedge\ \langle\tau\rangle(X \rhd \neg\mathit{UnitComp}))$$

which then implies

$$\mathit{UnitComp}\ \wedge\ X \mid \forall\, \alpha.[\alpha]((\exists\, w,u.\ \alpha = v(w,u)\ \wedge\ (\mathit{Idle}(v,n,u) \mid l\langle w,v\rangle))\ \vee\ (\alpha = \tau\ \wedge\ (\mathit{TermWaiter}(v,n) \mid l\langle n,v\rangle)))\ \wedge\ \langle\tau\rangle(X \rhd \neg\mathit{UnitComp})$$

by Lemma 4.10, which reduces to

$$\mathit{UnitComp}\ \wedge\ X \mid \langle\tau\rangle((\exists\, w,u.\ \tau = v(w,u)\ \wedge\ (\mathit{Idle}(v,n,u) \mid l\langle w,v\rangle))\ \vee\ (\tau = \tau\ \wedge\ (\mathit{TermWaiter}(v,n) \mid l\langle n,v\rangle)))\ \wedge\ (X \rhd \neg\mathit{UnitComp})$$

that is

$$\mathit{UnitComp}\ \wedge\ X \mid \langle\tau\rangle((\mathit{TermWaiter}(v,n) \mid l\langle n,v\rangle)\ \wedge\ (X \rhd \neg\mathit{UnitComp}))$$

and then

$$\langle\tau\rangle((\mathit{UnitComp} \mid \mathit{TermWaiter}(v,n) \mid l\langle n,v\rangle)\ \wedge\ \neg\mathit{UnitComp})$$

which implies $\langle\tau\rangle(\mathit{UnitComp} \wedge \neg\mathit{UnitComp})$, by applying Lemma 4.9 twice, that is $\langle\tau\rangle\mathbf{F}$, and then $\mathbf{F}$. $\square$

Lemma 5.5

$$\forall\ X.\ \forall\ v,n,l,q.\ \mathit{Spec}\ \wedge\ \mathit{UnitComp}\ \wedge\ (X \mid \mathit{Owner}(v,n,l,q)\ \wedge\ \langle\tau\rangle(X \triangleright \neg\mathit{UnitComp}))\ \Rightarrow\ \mathbf{F}$$

Proof. The formula in the antecedent implies

$$\mathit{Spec}\ \wedge\ (\mathit{UnitComp}\ \wedge\ X \mid \mathit{Owner}(v,n,l,q)\ \wedge\ \langle\tau\rangle(X \triangleright \neg\mathit{UnitComp}))$$

by Lemma 4.10, which then implies

$$\begin{array}{r}\mathit{UnitComp}\ \wedge\ X \mid \forall\ \alpha.[\alpha]((\exists\ w,u.\ \alpha = v(w,u)\ \wedge\ (\mathit{Owner}(v,n,u,q) \mid l\langle w,v\rangle))\ \vee \\ (\alpha = \tau\ \wedge\ (\mathit{Idle}(v,n,l) \mid q\langle\rangle)))\ \wedge\ \langle\tau\rangle(X \triangleright \neg\mathit{UnitComp})\end{array}$$

by Def. 4.1, which reduces to

$$\begin{array}{r}\mathit{UnitComp}\ \wedge\ X \mid \langle\tau\rangle((\exists\ w,u.\ \tau = v(w,u)\ \wedge\ (\mathit{Owner}(v,n,u,q) \mid l\langle w,v\rangle))\ \vee \\ (\tau = \tau\ \wedge\ (\mathit{Idle}(v,n,l) \mid q\langle\rangle)))\ \wedge\ (X \triangleright \neg\mathit{UnitComp})\end{array}$$

that is

$$\mathit{UnitComp}\ \wedge\ X \mid \langle\tau\rangle((\mathit{Idle}(v,n,l) \mid q\langle\rangle)\ \wedge\ (X \triangleright \neg\mathit{UnitComp}))$$

and then

$$\langle\tau\rangle((\mathit{UnitComp} \mid \mathit{Idle}(v,n,l) \mid q\langle\rangle)\ \wedge\ \neg\mathit{UnitComp})$$

which implies $\langle\tau\rangle(\mathit{UnitComp} \wedge \neg\mathit{UnitComp})$, by Lemma 4.9 , that is, $\langle\tau\rangle\mathbf{F}$, and then $\mathbf{F}$.
□

Lemma 5.6

$$\begin{array}{r}\forall\ X.\ \forall\ u,v,v_1,m,n.\ \mathit{Spec}\ \wedge\ \mathit{UnitComp}\ \wedge\ (X \mid v\langle m,u\rangle \mid \mathit{TermOwner}(v_1,n)\ \wedge \\ \langle v(m,u)\rangle(X \triangleright \neg\mathit{UnitComp}))\ \Rightarrow\ \mathbf{F}\end{array}$$

Proof. The antecedent implies

$$\mathit{Spec}\ \wedge\ (\mathit{UnitComp}\ \wedge\ X \mid v\langle m,u\rangle \mid \mathit{TermOwner}(v_1,n)\ \wedge\ \langle v(m,u)\rangle(X \triangleright \neg\mathit{UnitComp}))$$

by Lemma 4.10, and then we obtain

$$\begin{array}{r}\mathit{UnitComp}\ \wedge\ X \mid v\langle m,u\rangle \mid \forall\ \alpha.[\alpha](\exists\ w,u_1.\ \alpha = v_1(w,u_1)\ \wedge \\ \mathit{Owner}(v_1,n,u_1,w))\ \wedge \langle v(m,u)\rangle(X \triangleright \neg\mathit{UnitComp})\end{array}$$

by Def. 4.1, and then

$$\begin{array}{r}\mathit{UnitComp}\ \wedge\ X \mid v\langle m,u\rangle \mid\langle v(m,u)\rangle(\exists\ w,u_1.\ v(m,u) = v_1(w,u_1)\ \wedge \\ \mathit{Owner}(v_1,n,u_1,w)\ \wedge\ (X \triangleright \neg\mathit{UnitComp}))\end{array}$$

Now, if $v \neq v_1$ then the previous formula implies $X \mid v\langle m,u\rangle \mid \langle v(m,u)\rangle\mathbf{F}$, and so, it implies $\mathbf{F}$. Otherwise, if $v = v_1$ then

$$\mathit{UnitComp}\ \wedge\ X \mid \langle\tau\rangle(\mathit{Owner}(v,n,u,m)\ \wedge\ (X \triangleright \neg\mathit{UnitComp}))$$

by Axiom 2.6, which implies

$$\langle\tau\rangle((\mathit{UnitComp} \mid \mathit{Owner}(v,n,u,m))\ \wedge\ \neg\mathit{UnitComp})$$

Now, by Lemma 4.9, the previous formula implies $\langle\tau\rangle(\mathit{UnitComp} \wedge \neg\mathit{UnitComp})$, which implies $\langle\tau\rangle\mathbf{F}$, that is $\mathbf{F}$. □

Lemma 5.7

$$\forall\ X.\ \forall\ u, v, v_1, l, m, n.\ Spec\ \wedge\ UnitComp\ \wedge\ (X \mid v\langle m,u\rangle \mid Idle(v_1,n,l)\ \wedge \\ \langle v(m,u)\rangle(X \rhd \neg UnitComp))\ \Rightarrow\ \mathbf{F}$$

Proof. The antecedent implies

$$Spec\ \wedge\ (UnitComp\ \wedge\ X \mid v\langle m,u\rangle \mid Link(v_1,l)\ \wedge\ Node(v_1)\ \wedge\ Idle(v_1,n,l)\ \wedge \\ \langle v(m,u)\rangle(X \rhd \neg UnitComp))$$

by Lemma 4.10. Now, this formula implies

$$UnitComp\ \wedge\ X \mid v\langle m,u\rangle \mid \forall\ \alpha.[\alpha]((\exists\ w, u_1.\ \alpha = v_1(w,u_1)\ \wedge \\ (Idle(v_1,n,u_1) \mid l\langle w,v_1\rangle))\ \vee\ (\alpha = \tau\ \wedge\ (TermWaiter(v_1,n) \mid l\langle n,v_1\rangle)))\ \wedge \\ \langle v(m,u)\rangle(X \rhd \neg UnitComp)$$

by Def. 4.1, and which implies

$$UnitComp\ \wedge\ X \mid v\langle m,u\rangle \mid \langle v(m,u)\rangle(\exists\ w, u_1.\ v(m,u) = v_1(w,u_1)\ \wedge \\ (Idle(v_1,n,u_1) \mid l\langle w,v_1\rangle)\ \wedge\ (X \rhd \neg UnitComp))$$

Now, if $v \neq v_1$, then the previous formula implies $X \mid v\langle m,u\rangle \mid \langle v(m,u)\rangle\mathbf{F}$, that is, $\mathbf{F}$.

If $v = v_1$, we obtain

$$UnitComp\ \wedge\ X \mid \langle\tau\rangle((Idle(v,n,u) \mid l\langle m,v\rangle)\ \wedge\ (X \rhd \neg UnitComp))$$

by Axiom 2.6, and so

$$\langle\tau\rangle((UnitComp \mid Idle(v,n,u) \mid l\langle m,v\rangle)\ \wedge\ \neg UnitComp)$$

Now, by applying Lemma 4.9 twice, we obtain $\langle\tau\rangle(UnitComp \wedge \neg UnitComp)$, and so $\langle\tau\rangle\mathbf{F}$, which implies $\mathbf{F}$. □

5.3 Proof of *UniqueUndirPath*

Next we prove that $UniqueUndirPath$ is an invariant, which is obtained from the following two lemmas.

Lemma 5.8 $Init\ \Rightarrow\ UniqueUndirPath$

Proof. Let's assume $Init$ and $\neg UniqueUndirPath$, which implies

$$\exists\ x, y.\ \ x \neq y\ \wedge\ (?UndirPath(x,y) \mid ?UndirPath(x,y))$$

Then

$$\exists\ z, w.\ LinkedNode(x,z) \mid LinkedNode(x,w) \mid ?UndirPath(z,y) \mid ?UndirPath(w,y)$$

by expanding both the $?UndirPath(x,y)$ formulas, and by the result we have already proved in Lemma 5.1, that $Init\ \Rightarrow\ \bigstar(\mathbf{1} \Rightarrow Node)$, and so there are no links formed by request messages. Then, by Def. 4.1, every node is linked to at most another one, and so the previous formula implies $\mathbf{F}$. Then, $UniqueUndirPath$ follows from $Init$. □

In order to prove that from every state that satisfies $UniqueUndirPath$ we always reach a state that also satisfies it, we assume that $UnitComp$ is true, because we have already proved that this property is an invariant.

Lemma 5.9 $Spec \wedge UnitComp \wedge UniqueUndirPath \Rightarrow \Box UniqueUndirPath$

Proof. We prove it by contradiction. Assume that

$$Spec \wedge UnitComp \wedge UniqueUndirPath$$

and also that $\langle\rangle\neg UniqueUndirPath$. Now

$$\begin{array}{l} Spec \wedge UnitComp \wedge UniqueUndirPath \wedge \langle\rangle\neg UniqueUndirPath \Rightarrow \\ Spec \wedge UnitComp \wedge UniqueUndirPath \wedge (\exists\, m.(m\langle\rangle \mid \langle m()\rangle\neg UniqueUndirPath) \vee \\ \qquad \exists\, X.(X \mid \mathbf{1} \wedge \langle\tau\rangle(X \rhd \neg UniqueUndirPath))) \end{array}$$

by Axiom 2.6, which then implies the two cases

$$Spec \wedge UnitComp \wedge UniqueUndirPath \wedge \exists\, m.(m\langle\rangle \mid \langle m()\rangle\neg UniqueUndirPath) \quad (22)$$

and

$$Spec \wedge UnitComp \wedge UniqueUndirPath \wedge \exists\, X.(X \mid \mathbf{1} \wedge \langle\tau\rangle(X \rhd \neg UniqueUndirPath)) \quad (23)$$

Now, formula (22) implies

$$\begin{array}{l} Spec \wedge UnitComp \wedge UniqueUndirPath \wedge \\ \exists\, m.(m\langle\rangle \wedge Msg \mid \exists\, X.\ (X \mid \mathbf{1} \wedge \langle m()\rangle(X \rhd \neg UniqueUndirPath))) \end{array}$$

by *UnitComp* in Def. 4.6 and by Axiom 2.5, and then

$$\begin{array}{l} Spec \wedge UnitComp \wedge UniqueUndirPath \wedge \\ \exists\, m.\ (m\langle\rangle \wedge Msg \mid \exists\, X.\ (X \mid Node \wedge \langle m()\rangle(X \rhd \neg UniqueUndirPath))) \end{array}$$

also by *UnitComp*. Then, by *Msg* in Def. 4.2, the two cases follow

$$\begin{array}{r} Spec \wedge UnitComp \wedge UniqueUndirPath \wedge \exists\, obj.\ (obj\langle\rangle \mid \exists\, X.\ (X \mid \\ Node \wedge \langle obj()\rangle(X \rhd \neg UniqueUndirPath))) \end{array} \quad (24)$$

and

$$\begin{array}{r} Spec \wedge UnitComp \wedge UniqueUndirPath \wedge \exists\, find, move, from.\ (find\langle move, from\rangle \mid \\ \exists\, X.\ (X \mid Node \wedge \langle find(move, from)\rangle(X \rhd \neg UniqueUndirPath))) \end{array} \quad (25)$$

In the case that formula (24) is true, we obtain

$$\begin{array}{r} Spec \wedge UniqueUndirPath \wedge \exists\, X.\, \exists\, obj.\ (X \mid obj\langle\rangle \mid \\ (\exists\, find, move.\ TermOwner(find, move) \vee \\ \exists\, find, move.\ TermWaiter(find, move) \vee \\ \exists\, find, move, link.\ Idle(find, move, link) \vee \\ \exists\, find, move, link, queue.\ Waiter(find, move, link, queue) \vee \\ \exists\, find, move, link, queue.\ Owner(find, move, link, queue)) \wedge \\ \langle obj()\rangle(X \rhd \neg UniqueUndirPath)) \end{array}$$

by *Node* in Def. 4.2. Eliminating the existential quantifiers, the previous formula implies the five cases

$$\begin{array}{r} Spec \wedge UniqueUndirPath \wedge (X \mid obj\langle\rangle \mid TermOwner(find, move) \wedge \\ \langle obj()\rangle(X \rhd \neg UniqueUndirPath)) \end{array} \quad (26)$$

$$Spec \ \wedge \ UniqueUndirPath \ \wedge \ (X \mid obj\langle\rangle \mid TermWaiter(find, move) \ \wedge \\ \langle obj()\rangle(X \triangleright \neg UniqueUndirPath)) \tag{27}$$

$$Spec \ \wedge \ UniqueUndirPath \ \wedge \ (X \mid obj\langle\rangle \mid Idle(find, move, link) \ \wedge \\ \langle obj()\rangle(X \triangleright \neg UniqueUndirPath)) \tag{28}$$

$$Spec \ \wedge \ UniqueUndirPath \ \wedge \ (X \mid obj\langle\rangle \mid Waiter(find, move, link, queue) \ \wedge \\ \langle obj()\rangle(X \triangleright \neg UniqueUndirPath)) \tag{29}$$

$$Spec \ \wedge \ UniqueUndirPath \ \wedge \ (X \mid obj\langle\rangle \mid Owner(find, move, link, queue) \ \wedge \\ \langle obj()\rangle(X \triangleright \neg UniqueUndirPath)) \tag{30}$$

The formulas (26), (28) and (30) imply **F**, as it is illustrated next, for the case of formula (28)

$$\begin{array}{rr} (28) \quad \Rightarrow & X \mid obj\langle\rangle \mid \forall\, \alpha.[\alpha]((\exists\, ask, from.\ \alpha = find(ask, from) \wedge \\ & (Idle(find, move, from) \mid link\langle ask, find\rangle)) \ \vee \\ & (\alpha = \tau \wedge (TermWaiter(find, move) \mid link\langle ask, find\rangle))) \ \wedge \\ & \langle obj()\rangle(X \triangleright \neg UniqueUndirPath) \end{array}$$

Now, the previous formula implies

$$\begin{array}{r} X \mid obj\langle\rangle \mid \langle obj()\rangle((\exists\, ask, from.\ obj() = find(ask, from) \wedge \\ (Idle(find, move, from) \mid link\langle ask, find\rangle)) \ \vee \\ (obj() = \tau \wedge (TermWaiter(find, move) \mid link\langle ask, find\rangle))) \ \wedge \\ (X \triangleright \neg UniqueUndirPath) \end{array}$$

which implies $X \mid obj\langle\rangle \mid \langle obj()\rangle\mathbf{F}$, that is **F**.

Lemma 5.10 proves that also formula (27) implies **F**. The case of formula (29) implying **F** is proved in lemma 5.11.

Now, formula (25) implies

$$\begin{array}{r} Spec \ \wedge \ UniqueUndirPath \ \wedge \exists\, X.\ \exists\, find, ask, from.\ (X \mid find\langle ask, from\rangle \mid \\ (\exists\, find_1, move.\ TermOwner(find_1, move) \ \vee \\ \exists\, find_1, move.\ TermWaiter(find_1, move) \ \vee \\ \exists\, find_1, move, link.\ Idle(find_1, move, link) \ \vee \\ \exists\, find_1, move, link, queue.\ Waiter(find_1, move, link, queue) \ \vee \\ \exists\, find_1, move, link, queue.\ Owner(find_1, move, link, queue)) \ \wedge \\ \langle find(ask, from)\rangle(X \triangleright \neg UniqueUndirPath)) \end{array}$$

by $Node$ in Def. 4.2. Consequently, we obtain five cases

$$Spec \wedge UniqueUndirPath \wedge (X \mid find\langle ask, from\rangle \mid TermOwner(find_1, move) \wedge \\ \langle find(ask, from)\rangle(X \triangleright \neg UniqueUndirPath)) \tag{31}$$

$$Spec \wedge UniqueUndirPath \wedge (X \mid find\langle ask, from\rangle \mid TermWaiter(find_1, move) \wedge \\ \langle find(ask, from)\rangle(X \triangleright \neg UniqueUndirPath)) \tag{32}$$

$$Spec \wedge UniqueUndirPath \wedge (X \mid find\langle ask, from\rangle \mid Idle(find_1, move, link) \wedge \langle find(ask, from)\rangle(X \triangleright \neg UniqueUndirPath)) \quad (33)$$

$$Spec \wedge UniqueUndirPath \wedge (X \mid find\langle ask, from\rangle \mid Waiter(find_1, move, link, queue) \wedge \langle find(ask, from)\rangle(X \triangleright \neg UniqueUndirPath)) \quad (34)$$

$$Spec \wedge UniqueUndirPath \wedge (X \mid find\langle ask, from\rangle \mid Owner(find_1, move, link, queue) \wedge \langle find(ask, from)\rangle(X \triangleright \neg UniqueUndirPath)) \quad (35)$$

Formula (31) implies **F**, as proved in lemma 5.14; from formula (33) we also obtain a contradiction as proved in lemma 5.15; the proof of a contradiction for the other cases follow similarly to the previous cases, depending on the node being terminal or nonterminal, respectively.

Finally, the case of an internal transition, when formula (23) is satisfied, is considered. Formula (23) implies the five next cases, as usual:

$$Spec \wedge UniqueUndirPath \wedge (X \mid TermOwner(find, move) \wedge \langle \tau\rangle(X \triangleright \neg UniqueUndirPath)) \quad (36)$$

$$Spec \wedge UniqueUndirPath \wedge (X \mid TermWaiter(find, move) \wedge \langle \tau\rangle(X \triangleright \neg UniqueUndirPath)) \quad (37)$$

$$Spec \wedge UniqueUndirPath \wedge (X \mid Idle(find, move, link) \wedge \langle \tau\rangle(X \triangleright \neg UniqueUndirPath)) \quad (38)$$

$$Spec \wedge UniqueUndirPath \wedge (X \mid Waiter(find, move, link, queue) \wedge \langle \tau\rangle(X \triangleright \neg UniqueUndirPath)) \quad (39)$$

$$Spec \wedge UniqueUndirPath \wedge (X \mid Owner(find, move, link, queue) \wedge \langle \tau\rangle(X \triangleright \neg UniqueUndirPath)) \quad (40)$$

The formulas (36), (37) and (39) imply **F**, which can be proved similarly as we did for the case when formula (28) was true. The proof that formula (38) implies **F** is done in Lemma 5.12 and for formula (40) in Lemma 5.13. □

5.4 Auxiliary Lemmas for $UniqueUndirPath$

Lemma 5.10

$$\forall\, X.\, \forall\, v, m, n.\ Spec \wedge UniqueUndirPath \wedge (X \mid m\langle\rangle \mid TermWaiter(v, n) \wedge \langle m()\rangle(X \triangleright \neg UniqueUndirPath)) \Rightarrow \mathbf{F}$$

Proof. The formula in the antecedent implies

$$Spec \wedge UniqueUndirPath \wedge (UniqueUndirPath \wedge X \mid m\langle\rangle \mid TermWaiter(v, n) \wedge \langle m()\rangle(X \triangleright \neg UniqueUndirPath))$$

by Lemma 4.19, and then

$$\begin{aligned}&\mathit{UniqueUndirPath} \ \wedge \ X \mid m\langle\rangle \mid \forall\, \alpha.[\alpha]((\exists\, w, u.\ \alpha = v(w,u) \ \wedge \ \mathit{Waiter}(v,n,u,w)) \ \vee \\ &\qquad (\alpha = n() \ \wedge \ \mathit{TermOwner}(v,n))) \ \wedge \ \langle m()\rangle(X \triangleright \neg \mathit{UniqueUndirPath})\end{aligned}$$

by Def. 4.1, which reduces to

$$\begin{aligned}&\mathit{UniqueUndirPath} \ \wedge \ X \mid m\langle\rangle \mid \langle m()\rangle((\exists\, w, u.\ m() = v(w,u) \ \wedge \ \mathit{Waiter}(v,n,u,w)) \ \vee \\ &\qquad (m() = n() \ \wedge \ \mathit{TermOwner}(v,n))) \ \wedge \ (X \triangleright \neg \mathit{UniqueUndirPath})\end{aligned}$$

Now, if $m = n$, the formula reduces to

$$\mathit{UniqueUndirPath} \ \wedge \ X \mid \langle\tau\rangle(\mathit{TermOwner}(v,n) \ \wedge \ (X \triangleright \neg \mathit{UniqueUndirPath}))$$

by Axiom 2.6, and then

$$\langle\tau\rangle((\mathit{UniqueUndirPath} \mid \mathit{TermOwner}(v,n)) \ \wedge \ \neg \mathit{UniqueUndirPath})$$

by extending the scope of the transition, which implies

$$\langle\tau\rangle(\mathit{UniqueUndirPath} \wedge \neg \mathit{UniqueUndirPath})$$

by lemma 4.16, that is, **F**.

If $m \neq n$ then the formula implies $\langle\tau\rangle\mathbf{F}$, because both equalities are false, which implies **F**. □

Lemma 5.11

$$\begin{aligned}&\forall\, X.\ \forall\, v,l,m,n,q.\ \mathit{Spec} \ \wedge \ \mathit{UniqueUndirPath} \ \wedge \ (X \mid m\langle\rangle \mid \mathit{Waiter}(v,n,l,q) \ \wedge \\ &\qquad \langle m()\rangle(X \triangleright \neg \mathit{UniqueUndirPath})) \ \Rightarrow \ \mathbf{F}\end{aligned}$$

Proof. The formula in the antecedent implies

$$\begin{aligned}&\mathit{Spec} \ \wedge \ \mathit{UniqueUndirPath} \ \wedge \ (\mathit{UniqueUndirPath} \ \wedge \ X \mid m\langle\rangle \mid \\ &\qquad \mathit{Waiter}(v,n,l,q) \ \wedge \ \langle m()\rangle(X \triangleright \neg \mathit{UniqueUndirPath}))\end{aligned}$$

by Lemma 4.19, and then

$$\begin{aligned}&\mathit{Spec} \ \wedge \ (\mathit{UniqueUndirPath} \ \wedge \ X \ \wedge \ \neg ?\mathit{UndirPath}(v,l) \mid m\langle\rangle \mid \\ &\mathit{Waiter}(v,n,l,q) \ \wedge \ \mathit{UndirPath}(v,l) \ \wedge \ \langle m()\rangle(X \triangleright \neg \mathit{UniqueUndirPath}))\end{aligned}$$

by Lemma 4.18 and by the definition of *UndirPath*, that is

$$\begin{aligned}&\mathit{UniqueUndirPath} \ \wedge \ X \ \wedge \ \neg ?\mathit{UndirPath}(v,l) \mid m\langle\rangle \mid \\ &\forall\, \alpha.[\alpha]((\exists\, w, u.\ \alpha = v(w,u) \ \wedge \ (\mathit{Waiter}(v,n,u,q) \mid l\langle w,v\rangle)) \ \vee \\ &(\alpha = n() \ \wedge \ \mathit{Owner}(v,n,l,q))) \ \wedge \ \langle m()\rangle(X \triangleright \neg \mathit{UniqueUndirPath})\end{aligned}$$

by Def. 4.1, which reduces to

$$\begin{aligned}&\mathit{UniqueUndirPath} \ \wedge \ X \ \wedge \ \neg ?\mathit{UndirPath}(v,l) \mid m\langle\rangle \mid \\ &\langle m()\rangle((\exists\, w, u.\ m() = v(w,u) \ \wedge \ (\mathit{Waiter}(v,n,u,q) \mid l\langle w,v\rangle)) \ \vee \\ &(m() = n() \ \wedge \ \mathit{Owner}(v,n,l,q))) \ \wedge \ (X \triangleright \neg \mathit{UniqueUndirPath})\end{aligned}$$

Now, if $m = n$, the formula reduces to

$$\begin{aligned}&\mathit{UniqueUndirPath} \ \wedge \ X \ \wedge \ \neg ?\mathit{UndirPath}(v,l) \mid \\ &\langle\tau\rangle(\mathit{Owner}(v,n,l,q) \ \wedge \ (X \triangleright \neg \mathit{UniqueUndirPath}))\end{aligned}$$

by Axiom 2.6, and then

$$\langle\tau\rangle((UniqueUndirPath \ \wedge \ \neg?UndirPath(v,l) \mid Owner(v,n,l,q)) \ \wedge \\ \neg UniqueUndirPath)$$

which implies $\langle\tau\rangle(UniqueUndirPath \wedge \neg UniqueUndirPath)$ by Lemma 4.17, that is, $\mathbf{F}$.

If $m \neq n$ then the formula implies $\langle\tau\rangle\mathbf{F}$, that is $\mathbf{F}$. $\square$

Lemma 5.12

$$\forall \ X. \ \forall \ v,n,l. \ Spec \ \wedge \ UniqueUndirPath \ \wedge \ (X \mid Idle(v,n,l) \ \wedge \\ \langle\tau\rangle(X \triangleright \neg UniqueUndirPath)) \ \Rightarrow \ \mathbf{F}$$

Proof. The formula in the antecedent implies

$$Spec \ \wedge \ UniqueUndirPath \ \wedge \ (UniqueUndirPath \ \wedge \ X \mid \\ Idle(v,n,l) \ \wedge \ \langle\tau\rangle(X \triangleright \neg UniqueUndirPath))$$

by Lemma 4.19, which then implies

$$Spec \ \wedge \ (UniqueUndirPath \ \wedge \ X \ \wedge \ \neg?UndirPath(v,l) \mid \\ Idle(v,n,l) \ \wedge \ \langle\tau\rangle(X \triangleright \neg NoCyUnPath))$$

by Lemma 4.18, and then

$$UniqueUndirPath \ \wedge \ X \ \wedge \ \neg?UndirPath(v,l) \mid \\ \forall \ \alpha.[\alpha]((\exists \ w,u. \ \alpha = v(w,u) \ \wedge \ (Idle(v,n,u) \mid l\langle w,v\rangle)) \ \vee \\ (\alpha = \tau \ \wedge \ (TermWaiter(v,n) \mid l\langle n,v\rangle))) \ \wedge \ \langle\tau\rangle(X \triangleright \neg NoCyUnPath)$$

by Def 4.1, which reduces to

$$UniqueUndirPath \ \wedge \ X \ \wedge \ \neg?UndirPath(v,l) \mid \\ \langle\tau\rangle((\exists \ w,u. \ \tau = v(w,u) \ \wedge \ (Idle(v,n,u) \mid l\langle w,v\rangle)) \ \vee \\ (\tau = \tau \ \wedge \ (TermWaiter(v,n) \mid l\langle n,v\rangle))) \ \wedge \ (X \triangleright \neg?UniqueUndirPath)$$

that is

$$UniqueUndirPath \ \wedge \ X \ \wedge \ \neg?UndirPath(v,l) \mid \\ \langle\tau\rangle((TermWaiter(v,n) \mid l\langle n,v\rangle) \ \wedge \ (X \triangleright \neg UniqueUndirPath))$$

and then

$$\langle\tau\rangle((UniqueUndirPath \ \wedge \ \neg?UndirPath(v,l) \mid \\ TermWaiter(v,n) \mid l\langle n,v\rangle) \ \wedge \ \neg UniqueUndirPath)$$

which implies $\langle\tau\rangle((UniqueUndirPath \mid TermWaiter(v,n)) \wedge \neg UniqueUndirPath)$, by Lemma 4.17, and then, by Lemma 4.16, we obtain $\langle\tau\rangle(UniqueUndirPath \wedge \neg UniqueUndirPath)$ that is, $\langle\tau\rangle\mathbf{F}$, and then $\mathbf{F}$. $\square$

Lemma 5.13

$$\forall \ X. \ \forall \ v,n,l,q. \ Spec \ \wedge \ UniqueUndirPath \ \wedge \ (X \mid Owner(v,n,l,q) \ \wedge \\ \langle\tau\rangle(X \triangleright \neg UniqueUndirPath)) \ \Rightarrow \ \mathbf{F}$$

Proof. The formula in the antecedent implies

$$\begin{array}{c} Spec \ \wedge \ UniqueUndirPath \ \wedge \ (UniqueUndirPath \ \wedge \ X \mid \\ Owner(v,n,l,q) \ \wedge \ \langle\tau\rangle(X \triangleright \neg UniqueUndirPath)) \end{array}$$

by Lemma 4.19, which then implies

$$\begin{array}{c} Spec \ \wedge \ (UniqueUndirPath \ \wedge \ X \ \wedge \ \neg ?UniqPath(v,l) \mid \\ Owner(v,n,l,q) \ \wedge \ \langle\tau\rangle(X \triangleright \neg UniqueUndirPath)) \end{array}$$

by Lemma 4.18, and then

$$\begin{array}{c} UniqueUndirPath \ \wedge \ X \ \wedge \ \neg ?UndirPath(v,l) \mid \\ \forall\ \alpha.[\alpha]((\exists\ w,u.\ \alpha = v(w,u) \ \wedge \ (Owner(v,n,u,q) \mid l\langle w,v\rangle)) \ \vee \\ (\alpha = \tau \ \wedge \ (Idle(v,n,l) \mid q\langle\rangle))) \ \wedge \ \langle\tau\rangle(X \triangleright \neg UniqueUndirPath) \end{array}$$

by Def. 4.1, which reduces to

$$\begin{array}{c} UniqueUndirPath \ \wedge \ X \ \wedge \ \neg ?UndirPath(v,l) \mid \\ \langle\tau\rangle((\exists\ w,u.\ \tau = v(w,u) \ \wedge \ (Owner(v,n,u,q) \mid l\langle w,v\rangle)) \ \vee \\ (\tau = \tau \ \wedge \ (Idle(v,n,l) \mid q\langle\rangle))) \ \wedge \ (X \triangleright \neg UniqueUndirPath) \end{array}$$

that is

$$\begin{array}{c} UniqueUndirPath \ \wedge \ X \ \wedge \ \neg ?UndirPath(v,l) \mid \\ \langle\tau\rangle((Idle(v,n,l) \mid q\langle\rangle) \ \wedge \ (X \triangleright \neg UniqueUndirPath)) \end{array}$$

and then

$$\begin{array}{c} \langle\tau\rangle((UniqueUndirPath \ \wedge \ \neg ?UndirPath(v,l) \mid \\ Idle(v,n,l) \mid q\langle\rangle) \ \wedge \ \neg UniqueUndirPath) \end{array}$$

which implies $\langle\tau\rangle(UniqueUndirPath \wedge \neg UniqueUndirPath)$, by Lemma 4.17, that is, $\langle\tau\rangle\mathbf{F}$, and then $\mathbf{F}$. □

Lemma 5.14

$$\begin{array}{c} \forall\ X.\ \forall\ u,v,v_1,m,n.\ Spec \ \wedge \ UniqueUndirPath \ \wedge \ (X \mid v\langle m,u\rangle \mid \\ TermOwner(v_1,n) \ \wedge \ \langle v(m,u)\rangle(X \triangleright \neg UniqueUndirPath)) \ \Rightarrow \ \mathbf{F} \end{array}$$

Proof. The antecedent implies

$$\begin{array}{c} Spec \ \wedge \ UniqueUndirPath \ \wedge \ (X \mid UndirPath(u,v) \ \wedge \ v\langle m,u\rangle \mid \\ TermOwner(v_1,n) \ \wedge \ \langle v(m,u)\rangle(X \triangleright \neg UniqueUndirPath)) \end{array}$$

by the definition of *UndirPath*. Now, the previous formula implies

$$\begin{array}{c} Spec \ \wedge \ (UniqueUndirPath \ \wedge \ X \ \wedge \ \neg ?UndirPath(u,v) \mid v\langle m,u\rangle \mid \\ TermOwner(v_1,n) \ \wedge \ \langle v(m,u)\rangle(X \triangleright \neg UniqueUndirPath)) \end{array}$$

by Lemmas 4.18 and 4.19.

Now, by Def. 4.1, we obtain

$$\begin{array}{c} UniqueUndirPath \wedge X \wedge \neg ?UndirPath(u,v) \mid v\langle m,u\rangle \mid \\ \forall\ \alpha.[\alpha](\exists\ w,u_1.\ \alpha = v_1(w,u_1) \ \wedge \ Owner(v_1,n,u_1,w)) \wedge \\ \langle v(m,u)\rangle(X \triangleright \neg UniqueUndirPath) \end{array}$$

and then

$$\begin{array}{c} UniqueUndirPath \wedge X \wedge \neg ?UndirPath(v,u) \mid v\langle m,u\rangle \mid \\ \langle v(m,u)\rangle(\exists\ w,u_1.\ v(m,u) = v_1(w,u_1)\ \wedge\ Owner(v_1,n,u_1,w)\ \wedge \\ \hfill (X \triangleright \neg UniqueUndirPath)) \end{array}$$

Now, if $v \neq v_1$ then the previous formula implies $X \mid v\langle m,u\rangle \mid \langle v(m,u)\rangle\mathbf{F}$, and so, it implies $\mathbf{F}$. Otherwise, if $v = v_1$ then

$$\begin{array}{c} UniqueUndirPath \wedge X \wedge \neg ?UndirPath(u,v) \mid \\ \langle\tau\rangle(Owner(v,n,u,m)\ \wedge\ (X \triangleright \neg UniqueUndirPath)) \end{array}$$

by Axiom 2.6, which implies

$$\begin{array}{c} \langle\tau\rangle((UniqueUndirPath\ \wedge\ \neg ?UndirPath(u,v) \mid Owner(v,n,u,m))\ \wedge \\ \hfill \neg UniqueUndirPath) \end{array}$$

Now, by Lemma 4.17, the previous formula implies

$$\langle\tau\rangle(UniqueUndirPath\ \wedge\ \neg UniqueUndirPath)$$

that is, $\mathbf{F}$. $\Box$

Lemma 5.15

$$\begin{array}{c} \forall\ X.\ \forall\ u,v,v_1,l,m,n.\ Spec\ \wedge\ UniqueUndirPath\ \wedge\ (X \mid v\langle m,u\rangle \mid \\ Idle(v_1,n,l)\ \wedge\ \langle v(m,u)\rangle(X \triangleright \neg UniqueUndirPath))\ \Rightarrow\ \mathbf{F} \end{array}$$

Proof. The antecedent implies

$$\begin{array}{c} UniqueUndirPath\ \wedge\ (UniqueUndirPath\ \wedge\ X \mid v\langle m,u\rangle \mid \\ \forall\ \alpha.[\alpha]((\exists\ w,u_1.\ \alpha = v_1(w,u_1)\ \wedge\ (Idle(v_1,n,u_1) \mid l\langle w,v_1\rangle))\ \vee \\ (\alpha = \tau\ \wedge\ (TermWaiter(v_1,n) \mid l\langle n,v_1\rangle)))\ \wedge\ UndirPath(v_1,l)\ \wedge \\ \hfill \langle v(m,u)\rangle(X \triangleright \neg UniqueUndirPath)) \end{array}$$

by Def. 4.1, by the definition of $UndirPath$ and by Lemma 4.19, which implies

$$\begin{array}{c} UniqueUndirPath\ \wedge\ X \mid UndirPath(u,v)\ \wedge\ v\langle m,u\rangle \mid UndirPath(v_1,l)\ \wedge \\ \langle v(m,u)\rangle(\exists\ w,u_1.\ v(m,u) = v_1(w,u_1)\ \wedge\ (Idle(v_1,n,u_1) \mid l\langle w,v_1\rangle)\ \wedge \\ \hfill (X \triangleright \neg UniqueUndirPath)) \end{array}$$

Now, if $v \neq v_1$, then the previous formula implies $X \mid v\langle m,u\rangle \mid \langle v(m,u)\rangle\mathbf{F}$, that is, $\mathbf{F}$.

If $v = v_1$, we obtain

$$\begin{array}{c} UniqueUndirPath\ \wedge\ X\ \wedge\ \neg ?UndirPath(u,l) \mid v\langle m,u\rangle \mid \\ \langle v(m,u)\rangle((Idle(v,n,u) \mid l\langle m,v\rangle)\ \wedge\ (X \triangleright \neg UniqueUndirPath)) \end{array}$$

by Lemma 4.18, after the application of Lemma 4.14 to deduce that $?UnlinkPath(u,l)$ for the last two components. Then, we obtain

$$\begin{array}{c} UniqueUndirPath\ \wedge\ X\ \wedge\ \neg ?UndirPath(u,l) \mid \\ \langle\tau\rangle((Idle(v,n,u) \mid l\langle m,v\rangle)\ \wedge\ (X \triangleright \neg UniqueUndirPath)) \end{array}$$

by Axiom 2.6, and so

$$\begin{array}{c} \langle\tau\rangle((UniqueUndirPath\ \wedge\ \neg ?UndirPath(u,l) \mid (Idle(v,n,u) \mid l\langle m,v\rangle)\ \wedge \\ \hfill ?UndirPath(u,l))\ \wedge\ \neg UniqueUndirPath) \end{array}$$

and, then by Lemma 4.17, we obtain

$$\langle\tau\rangle(UniqueUndirPath\ \wedge\ \neg UniqueUndirPath)$$

that is, $\langle\tau\rangle\mathbf{F}$, which implies $\mathbf{F}$. $\Box$

5.5 Proof of $Safety$

It remains to prove the main invariant, the property $Safety$. Now, that we have already proved that $UniqueUndirPath$ is a safety property, we prove that it implies $Safety$ and consequently, this one is a safety property too.

Lemma 5.16 $UniqueUndirPath \;\Rightarrow\; Safety$

Proof. Assume $UniqueUndirPath$. We have to prove that, for all x such that $?Node(x)$, also $?Target(x)$. If $?TerminalNode(x)$ then also $?Target(x)$, by definition. Otherwise, $\exists\ y.\ LinkedNode(x,y)$. Now, by the induction hypothesis $?Target(y)$ is true, so we obtain $LinkedNode(x,y) \;\wedge\; ?Target(y)$. In order to prove $?Target(x)$, we must prove that $LinkedNode(x,y) \mid ?Target(y)$. By Lemma 4.11, if $?Node(x) \wedge Target(y)$, then

$$PhyPath(y,x) \mid \exists\ z.\ PhyPath(x,z) \;\wedge\; ?Terminal(z)$$

but then

$$?LinkedNode(x,y) \;\wedge\; ?(PhyPath(y,x) \mid \exists\ z.\ PhyPath(x,z) \;\wedge\; ?Terminal(z))$$

which contradicts the hypothesis of $UniqueUndirPath$, and then $?Node(x)$ can not be true in the component satisfying $Target(y)$, and so $LinkedNode(x,y) \mid ?Target(y)$, that is, $Target(x)$. $\Box$

6 Concluding Remarks

In this report, we have developed a preliminary study of proof theoretic proof techniques for spatial logic specifications of distributed systems, using a P2P distributed protocol as a case study. This work already provides some insight about the structure of spatial logic proofs in a more applied setting, and suggests some specific techniques of general interest.

For example, the technique to prove an invariant given the system's specification follows the usual coinductive technique, but in the current setting, which is based on spatial logic, we have been able to derive the effect of a global transition of the system from the local specifications of the behavior of its parts. The fact that the reasoning process is conducted at various structural levels, in particular that spatial operators allow us to express local properties, such as properties related to local change in the system, seems to be an effective way of avoiding the use of complex global invariants. Building on this experience, we expect in future work to be able to isolate general proof techniques, possibly leading to partially automated reasoning principles, and consider distributed systems where restricted names and channels plays an important role.

Acknowledgments This work is partially supported by Profundis FET IST-2001-33100.

References

[1] L. Caires. Behavioral and spatial observations in a logic for the π-calculus. In *FOSSACS 2004: Foundations of Software Science and Computation Structures*, volume 2987 of *Lecture Notes in Computer Science*. Springer-Verlag, 2004.

[2] L. Caires and L. Cardelli. A spatial logic for concurrency (part ii). In *CONCUR 2002: Concurrency Theory (13th International Conference)*, volume 2421 of *Lecture Notes in Computer Science*, pages 209–225. Springer-Verlag, 2002.

[3] L. Caires and L. Cardelli. A spatial logic for concurrency (part i). In *Information and Computation*, volume 186(2), 2003.

[4] L. Cardelli and A. D. Gordon. Anytime, anywhere. modal logics for mobile ambients. In *27th ACM Symp. on Principles of Programming Languages*, volume 2987, pages 365–377. ACM, 2000.

[5] L. Cardelli and A. D. Gordon. Logical properties of name restriction. In S. Abramsky, editor, *Typed Lambda Calculi and Applications*, volume 2044. Springer-Verlag, 2001.

[6] M. Dam. Proof systems for pi-calculus logics. In *Logic for Concurrency and Synchronisation, R. de Queiroz (ed.)*, Trends in Logic, Studia Logica Library, pages 145–212. Kluwer, 2003.

[7] Michael J. Demmer and Maurice Herlihy. The arrow distributed directory protocol. In *Proceedings of the 12th International Symposium on Distributed Computing*, pages 119–133. Springer-Verlag, 1998.

[8] R. Milner. *Communicating and Mobile Systems: the π-calculus*. Cambridge University Press, 1999.

[9] J. Parrow R. Milner and D. Walker. Modal logics for mobile processes. *Theoretical Computer Science*, 114:149–171, 1993.