Epistemic Protocols for Distributed Gossiping

Krzysztof R. Apt
Centrum Wiskunde & Informatica
Amsterdam, The Netherlands
k.r.apt@cwi.nl

Davide Grossi
University of Liverpool
Liverpool, UK
d.grossi@liverpool.ac.uk

Wiebe van der Hoek
University of Liverpool
Liverpool, UK
wiebe@liv.ac.uk

ABSTRACT

Gossip protocols aim at arriving, by means of point-to-point or group communications, at a situation in which all the agents know each other's secrets. We consider distributed gossip protocols which are expressed by means of epistemic logic. We provide an operational semantics of such protocols and set up an appropriate framework to argue about their correctness. Then we analyze specific protocols for complete graphs and for directed rings.

General Terms

Theory

Keywords

Gossip protocols, epistemic logic, distributed computing, knowledge-based programs

1. INTRODUCTION

In the gossip problem ([18, 4], see also [10] for an overview) a number n of agents, each one knowing a piece of information (a *secret*) unknown to the others, communicate by one-to-one interactions (e.g., telephone calls). The result of each call is that the two agents involved in it learn all secrets the other agent knows at the time of the call. The problem consists in finding a sequence of calls which disseminates all the secrets among the agents in the group. It sparked a large literature in the 70s and 80s [18, 4, 9, 5, 17] typically focusing on establishing—in the above and other variants of the problem—the minimum number of calls to achieve dissemination of all the secrets. This number has been proven to be $2n - 4$, where n, the number of agents, is at least 4.

The above literature assumes a centralized perspective on the gossip problem: a planner schedules agents' calls. In this paper we pursue a line of research first put forth in [3] by developing a decentralized theory of the gossip problem, where agents perform calls not according to a centralized schedule, but following individual epistemic protocols they run in a distributed fashion. These protocols tell the agents which calls to execute depending on what they know, or do not know, about the information state of the agents in the group. We call the resulting distributed programs *(epistemic) gossip protocols*.

TARK 2015

Contribution of the paper and outline.

The paper introduces a formal framework for specifying epistemic gossip protocols and for studying their computations in terms of correctness, termination, and fair termination (Section 2). It then defines and studies two natural protocols in which the interactions are unconstrained (Section 3) and four example gossip protocols in which agents are positioned on a directed ring and calls can happen only between neighbours (Section 4). Proofs are collected in the appendix.

From a methodological point of view, the paper integrates concepts and techniques from the distributed computing, see, e.g., [1, Chapter 11] and the epistemic logic literature [8, 15] in the tradition of [16, 14, 7].

2. GOSSIP PROTOCOLS

We introduce first the syntax and semantics of gossip protocols.

2.1 Syntax

We loosely use the syntax of the language CSP (Communicating Sequential Processes) of [11] that extends the guarded command language of [6] by disjoint parallel composition and commands for synchronous communication. CSP was realized in the distributed programming language OCCAM (see INMOS [12]).

The main difference is that we use as guards epistemic formulas and as communication primitives calls that do not require synchronization. Also, the syntax of our distributed programs is very limited. In order to define gossip protocols we introduce in turn calls and epistemic guards.

Throughout the paper we assume a fixed finite set A of at least three ***agents***. We assume that each agent holds exactly one ***secret*** and that there exists a bijection between the set of agents and the set of secrets. We denote by P the set of all secrets (for *p*ropositions). Furthermore, it is assumed that each secret carries information identifying the agent to whom that secret belongs.

2.1.1 Calls

Each ***call*** concerns two agents, the *caller* (a below) and the *agent called* (b). We distinguish three ***modes of communication*** of a call:

push-pull, written as ab or (a, b). During this call the caller and the called agent learn each other's secrets,

push, written as $a \triangleright b$. After this call the called agent learns all the secrets held by the caller,

pull, written as $a \triangleleft b$. After this call the caller learns all the secrets held by the called agent.

Variables for calls are denoted by c, d. Abusing notation we write $a \in \mathsf{c}$ to denote that agent a is one of the two agents involved in the call c (e.g., for $\mathsf{c} := ab$ we have $a \in \mathsf{c}$ and $b \in \mathsf{c}$). Calls in which agent a is involved are denoted by c^a.

2.1.2 Epistemic guards

Epistemic guards are defined as formulas in a simple modal language with the following grammar:

$$\phi ::= F_a p \mid \neg\phi \mid \phi \wedge \phi \mid K_a \phi,$$

where $p \in \mathsf{P}$ and $a \in \mathsf{A}$. Each secret is viewed as a distinct symbol. We denote the secret of agent a by A, the secret of agent b by B and so on. We denote the set of so defined formulas by $\mathcal{L}$ and we refer to its members as epistemic formulas or epistemic guards. We read $F_a p$ as 'agent a is familiar with the secret p' (or 'p belongs to the set of secrets a knows about') and $K_a \phi$ as 'agent a knows that formula ϕ is true'. So this language is an epistemic language where atoms consist of 'knowing whether' statements about propositional atoms, if we view secrets as Boolean variables.

Atomic expressions in $\mathcal{L}$ concern only who knows what secrets. As a consequence the language cannot express formally the truth of a secret p. This level of abstraction suffices for the purposes of the current paper. However, expressions $F_a p$ could be given a more explicit epistemic reading in terms of 'knowing whether'. That is, 'a is familiar with p' can be interpreted (on a suitable Kripke model) as 'a knows whether the secret p is true or not'. This link is established in [3].

2.1.3 Gossip protocols

Before specifying what a program for agent a is, let us first define the language $\mathcal{L}_a$ with the following grammar:

$$\psi ::= K_a \phi \mid \neg\psi \mid \psi \wedge \psi$$

with $\phi \in \mathcal{L}$.[1]

By a ***component program***, in short a ***program***, for an agent a we mean a statement of the form

$$*[[]_{j=1}^{m}\ \psi_j \to \mathsf{c}_j],$$

where $m > 0$ and each $\psi_j \to \mathsf{c}_j$ is such that $\psi_j \in \mathcal{L}_a$ and a is the caller in c_j.

Given an epistemic formula $\psi \in \mathcal{L}_a$ and a call c, we call the construct $\psi \to \mathsf{c}$ a ***rule*** and refer in this context to ψ as a ***guard***.

We denote the set of rules $\{\psi_1 \to \mathsf{c}_1, \ldots, \psi_k \to \mathsf{c}_k\}$ as $[[]_{j=1}^{k}\ \psi_j \to \mathsf{c}_j]$ and abbreviate a set of rules $\{\psi_1 \to \mathsf{c}, \ldots, \psi_k \to \mathsf{c}\}$ with the same call to a single rule $\bigvee_{i=1}^{k} \psi_i \to \mathsf{c}$.

Intuitively, $*$ denotes a repeated execution of the rules, one at a time, where each time a rule is selected whose guard is true.

Finally, by a ***distributed epistemic gossip protocol***, in short a ***gossip protocol***, we mean a parallel composition of component programs, one for each agent. In order not to complicate matters we assume that each gossip protocol uses only one mode of communication.

Of special interest for this paper are gossip protocols that are symmetric. By this we mean that the protocol is a composition of the component programs that are identical modulo the names of the agents. Formally, consider a statement $\pi(x)$, where x is a variable ranging over the set A of agents and such that for each agent $a \in \mathsf{A}$, $\pi(a)$ is a component program for agent a. Then the parallel composition of the $\pi(a)$ programs, where $a \in \mathsf{A}$, is called a ***symmetric gossip protocol***.

Gossip protocols are syntactically extremely simple. Therefore it would seem that little can be expressed using them. However, this is not the case. In Sections 3 and 4 we consider gossip protocols that can exhibit complex behaviour.

2.2 Semantics

We now move on to provide a formal semantics of epistemic guards, and then describe the computations of gossip protocols.

2.2.1 Gossip situations and calls

A ***gossip situation*** is a sequence $\mathsf{s} = (\mathsf{Q}_a)_{a \in \mathsf{A}}$, where $\mathsf{Q}_a \subseteq \mathsf{P}$ for each agent a. Intuitively, Q_a is the set of secrets a is familiar with in situation s. The ***initial gossip situation*** is the one in which each Q_a equals $\{A\}$ and is denoted by root. The set of all gossip situations is denoted by S. We say that an agent a is an ***expert*** in a gossip situation s if he is familiar in s with all the secrets, i.e., if $\mathsf{Q}_a = \mathsf{P}$. The initial gossip situation reflects the fact that initially each agent is familiar only with his own secret, although it is not assumed this is common knowledge among the agents. In fact, in the introduced language we have no means to express the concept of common knowledge.

We will use the following concise notation for gossip situations. Sets of secrets will be written down as lists. e.g., the set $\{A, B, C\}$ will be written as ABC. Gossip situations will be written down as lists of lists of secrets separated by dots. E.g., if there are three agents, $\mathsf{root} = A.B.C$ and the situation $(\{A, B\}, \{A, B\}, \{C\})$ will be written as $AB.AB.C$.

Each call transforms the current gossip situation by modifying the set of secrets the agents involved in the call are familiar with. More precisely, the application of a call to a situation is defined as follows.

DEFINITION 2.1 (EFFECTS OF CALLS). *A call is a function* $\mathsf{c} : \mathsf{S} \longrightarrow \mathsf{S}$*, so defined, for* $\mathsf{s} := (\mathsf{Q}_a)_{a \in \mathsf{A}}$*:*

$\boxed{\mathsf{c} = ab}$ $\mathsf{c}(\mathsf{s}) = (Q'_a)_{a \in \mathsf{A}}$*, where* $\mathsf{Q}'_a = \mathsf{Q}'_b = \mathsf{Q}_a \cup \mathsf{Q}_b$*,* $\mathsf{Q}'_c = \mathsf{Q}_c$*, for* $c \neq a, b$*;*

$\boxed{\mathsf{c} = a \triangleright b}$ $\mathsf{c}(\mathsf{s}) = (Q'_a)_{a \in \mathsf{A}}$*, where* $\mathsf{Q}'_b = \mathsf{Q}_a \cup \mathsf{Q}_b$*,* $\mathsf{Q}'_a = \mathsf{Q}_a$*,* $\mathsf{Q}'_c = \mathsf{Q}_c$*, for* $c \neq a, b$*;*

$\boxed{\mathsf{c} = a \triangleleft b}$ $\mathsf{c}(\mathsf{s}) = (Q'_a)_{a \in \mathsf{A}}$*, where* $\mathsf{Q}'_a = \mathsf{Q}_a \cup \mathsf{Q}_b$*,* $\mathsf{Q}'_b = \mathsf{Q}_b$*,* $\mathsf{Q}'_c = \mathsf{Q}_c$*, for* $c \neq a, b$*.*

The definition formalizes the modes of communications we introduced earlier. Depending on the mode, secrets are either shared between caller and callee (ab), they are pushed from the caller to the callee ($a \triangleright b$), or they are retrieved by the caller from the callee ($a \triangleleft b$).

2.2.2 Call sequences

A ***call sequence*** is a (possibly infinite) sequence of calls, in symbols $(\mathsf{c}_1, \mathsf{c}_2, \ldots, \mathsf{c}_n, \ldots)$, all being of the same communication mode. The empty sequence is denoted by ϵ. We use $\mathbf{c}$ to denote a call sequence and $\mathbf{C}$ to denote the set of all call sequences. The set of all finite call sequences is denoted $\mathbf{C}^{<\omega}$. Given a finite call sequence $\mathbf{c}$ and a call c we denote by $\mathsf{c}.\mathbf{c}$ the prepending of $\mathbf{c}$ with c, and by $\mathbf{c}.\mathsf{c}$ the postpending of $\mathbf{c}$ with c.

The result of applying a call sequence to a situation s is defined by induction using Definition 2.1, as follows:

[1] Alternatively, $\mathcal{L}_a$ could be defined as the fragment of $\mathcal{L}$ consisting of the formulae of form $K_a \psi$. In logic S5, it is easy to prove that each $\psi \in \mathcal{L}_a$ is logically equivalent to a formula $K_a \phi \in \mathcal{L}$.

[Base] $\epsilon(\mathsf{s}) := \mathsf{s}$,
[Step] $(\mathsf{c}.\mathbf{c})(\mathsf{s}) := \mathbf{c}(\mathsf{c}(\mathsf{s}))$.

EXAMPLE 2.2. *Let the set of agents be* $\{a, b, c\}$.

$$\begin{array}{cccc} & ab & ca & ab \\ A.B.C & AB.AB.C & ABC.AB.ABC & ABC.ABC.ABC \end{array}$$

The top row lists the call sequence (ab, ca, ab)*, while the bottom row lists the successive gossip situations obtained from the initial situation* $A.B.C$ *by applying the calls in the sequence: first* ab*, then* ca *and finally* ab*.* □

By applying an infinite call sequence $\mathbf{c} = (\mathsf{c}_1, \mathsf{c}_2, \ldots, \mathsf{c}_n, \ldots)$ to a gossip situation s one obtains therefore an infinite sequence $\mathbf{c}^0(\mathsf{s}), \mathbf{c}^1(\mathsf{s}), \ldots, \mathbf{c}^n(\mathsf{s}), \ldots$ of gossip situations, where each $\mathbf{c}^k$ is sequence $\mathsf{c}_1, \mathsf{c}_2, \ldots, \mathsf{c}_k$. A call sequence $\mathbf{c}$ is said to ***converge*** if for all input gossip situations s the generated sequence of gossip situations reaches a limit, that is, there exists $n < \omega$ such that for all $m \geq n$ $\mathbf{c}^m(\mathsf{s}) = \mathbf{c}^{m+1}(\mathsf{s})$. Since the set of secrets is finite and calls never make agents forget secrets they are familiar with, it is easy to see the following.

FACT 2.3. *All infinite call sequences converge.*

However, as we shall see, this does not imply that all gossip protocols terminate. In the remainder of the paper, unless stated otherwise, we will assume the push-pull mode of communication. The reader can easily adapt our presentation to the other modes.

2.2.3 Gossip models

The set S of all gossip situations is the set of all possible combinations of secret distributions among the agents. As calls progress in sequence from the initial situation, agents may be uncertain about which one of such secrets distributions is the actual one. This uncertainty is precisely the object of the epistemic language for guards we introduced earlier.

DEFINITION 2.4. *A* **gossip model** *(for a given set* $\mathsf{A}*) is a tuple* $\mathcal{M} = (\mathbf{C}^{<\omega}, \{\sim_a\}_{a\in\mathsf{A}})$*, where each* $\sim_a \subseteq \mathbf{C}^{<\omega} \times \mathbf{C}^{<\omega}$ *is the smallest relation satisfying the following inductive conditions (assume the mode of communication is push-pull):*

[Base] $\epsilon \sim_a \epsilon$*;*

[Step] *Suppose* $\mathbf{c} \sim_a \mathbf{d}$*.*

(i) If $a \notin \mathsf{c}$*, then* $\mathbf{c}.\mathsf{c} \sim_a \mathbf{d}$ *and* $\mathbf{c} \sim_a \mathbf{d}.\mathsf{c}$*.*

(ii) If there exists $b \in \mathsf{A}$ *and* $\mathsf{c}, \mathsf{d} \in \{ab, ba\}$ *such that* $\mathbf{c}.\mathsf{c}(\mathsf{root})_a = \mathbf{d}.\mathsf{d}(\mathsf{root})_a$*, then* $\mathbf{c}.\mathsf{c} \sim_a \mathbf{d}.\mathsf{d}$*.*

A gossip model with a designated finite call sequence is called a **pointed gossip model**.

For the push, respectively pull, modes of communication clause (ii) needs to be modified by requiring that for some $b \in \mathsf{A}$*,* $\mathsf{c} = \mathsf{d} = a \triangleright b$ *or* $\mathsf{c} = \mathsf{d} = a \triangleleft b$*, respectively.*

For instance, by *(i)* we have $ab, bc \sim_a ab, bd$. But we do not have $bc, ab \sim_a bd, ab$ since $(bc, ab)(\mathsf{root})_a = ABC \neq ABD = (bd, ab)(\mathsf{root})_a$.

Let us flesh out the intuitions behind the above definition. Gossip models are needed in order to interpret the epistemic guards of gossip protocols. Since such guards are relevant only after finite sequences of calls, the domain of a gossip model is taken to consist only of finite sequences. Intuitively, those are the finite sequences that can be generated by a gossip protocol. Let us turn now to the $\sim_a$ relation. This is defined with the following intuitions in mind. First of all, no agent can distinguish the empty call sequence from itself—this is the base of the induction. Next, if two call sequences are indistinguishable for a, then the same is the case if *(i)* we extend one of these sequences by a call in which a is not involved or if *(ii)* we extend each of these sequences by a call of a with the same agent (agent a may be the caller or the callee), provided a is familiar with exactly the same secrets after each of the new sequences has taken place—this is the induction step.[2]

The above intuitions are based on the following assumptions on the form of communication we presuppose: (i) At the initial situation, as communication starts, each agent knows only her own secret but considers it possible that the others may be familiar with all other secrets. In other words there is no such thing as common knowledge of the fact that 'everybody knows exactly her own secret'. (ii) In general, each agent always considers it possible that call sequences (of any length) take place that do not involve her. These assumptions are weaker than the ones analyzed in [3].

We state without proof the following simple fact.

FACT 2.5.

(i) Each $\sim_a$ *is an equivalence relation;*

(ii) For all $\mathbf{c}, \mathbf{d} \in \mathbf{C}$ *if* $\mathbf{c} \sim_a \mathbf{d}$*, then* $\mathbf{c}(\mathsf{root})_a = \mathbf{d}(\mathsf{root})_a$*, but not vice versa.*

This prompts us to note also that according to Definition 2.4 sequences which make a learn the same set of secrets may well be distinguishable for a, such as, for instance, ab, bc, ab and ab, bc, ac. In the first one a comes to know that b knows a is familiar with all secrets, while in the second one, she comes to know that c knows a is familiar with all secrets. Relation $\sim_a$ is so defined as to capture this sort of 'higher-order' knowledge.

2.2.4 Truth conditions for epistemic guards

Everything is now in place to define the truth of the considered formulas.

DEFINITION 2.6. *Let* $(\mathcal{M}, \mathbf{c})$ *be a pointed gossip model with* $\mathcal{M} = (\mathbf{C}^{<\omega}, (\sim_a)_{a\in\mathsf{A}})$ *and* $\mathbf{c} \in \mathbf{C}^{<\omega}$*. We define the satisfaction relation* $\models$ *inductively as follows (clauses for Boolean connectives are omitted):*

$$\begin{array}{ll} (\mathcal{M}, \mathbf{c}) \models F_a p & \textit{iff} \quad p \in \mathbf{c}(\mathsf{root})_a, \\ (\mathcal{M}, \mathbf{c}) \models K_a \phi & \textit{iff} \quad \forall \mathbf{d} \textit{ s.t. } \mathbf{c} \sim_a \mathbf{d},\ (\mathcal{M}, \mathbf{d}) \models \phi. \end{array}$$

So formula $F_a p$ is true (in a pointed gossip model) whenever secret p belongs to the set of secrets agent a is familiar with in the situation generated by the designated call sequence $\mathbf{c}$ applied to the initial situation root. The knowledge operator is interpreted as customary in epistemic logic using the equivalence relations $\sim_a$.

2.2.5 Computations

Assume a gossip protocol P that is a parallel composition of the component programs $*[[]_{j=1}^{m_a}\ \psi_j^a \to \mathsf{c}_j^a]$, one for each agent $a \in \mathsf{A}$.

Given the gossip model $\mathcal{M} = (\mathbf{C}^{<\omega}, \{\sim_a\}_{a\in\mathsf{A}})$ we define the ***computation tree*** $\mathbf{C}^P \subseteq \mathbf{C}^{<\omega}$ of P as the smallest set of sequences satisfying the following inductive conditions:

[Base] $\epsilon \in \mathbf{C}^P$;

[2] Notice that the definition requires a designated initial situation, which we assume to be root.

[Step] If $\mathbf{c} \in \mathbf{C}^P$ and $(\mathcal{M}, \mathbf{c}) \models \psi_j^a$ then $\mathbf{c}.\mathsf{c}_j^a \in \mathbf{C}^P$. In this case we say that a ***transition*** has taken place between $\mathbf{c}$ and $\mathbf{c}.\mathsf{c}_j^a$, in symbols, $\mathbf{c} \to \mathbf{c}.\mathsf{c}_j^a$.

So $\mathbf{C}^P$ is a (possibly infinite) set of finite call sequences that is iteratively obtained by performing a 'legal' call (according to protocol P) from a 'legal' (according to protocol P) call sequence.

A ***path*** in the computation tree of P is a (possibly infinite) sequence of elements of $\mathbf{C}^P$, denoted by $\xi = (\mathbf{c}_0, \mathbf{c}_1, \ldots, \mathbf{c}_n, \ldots)$, where $\mathbf{c}_0 = \epsilon$ and each $\mathbf{c}_{i+1} = \mathbf{c}_i.\mathsf{c}$ for some call c and $i \geq 0$. A ***computation*** of P is a maximal rooted path in the computation tree of P.[3]

The above definition implies that a call sequence $\mathbf{c}$ is a leaf of the computation tree if and only if

$$(\mathcal{M}, \mathbf{c}) \models \bigwedge_{a \in \mathsf{A}} \bigwedge_{j=1}^{m_a} \neg \psi_j^a.$$

We call the formula

$$\bigwedge_{a \in \mathsf{A}} \bigwedge_{j=1}^{m_a} \neg \psi_j^a$$

the ***exit condition*** of the gossip protocol P.

Obviously computation trees can be infinite, though they are always finitely branching. Further, note that this semantics for gossip protocols abstracts away from some implementation details of the calls. More specifically, we assume that the caller always succeeds in his call and does not require to synchronize with the called agent. In reality, the called agent might be busy, being engaged in another call. To take care of this one could modify each call by replacing it by a 'call protocol' that implements the actual call using some lower level primitives. We do not elaborate further on this topic.

Let us fix some more terminology. For $\mathbf{c} \in \mathbf{C}^P$, an agent a is ***enabled*** in $\mathbf{c}$ if $(\mathcal{M}, \mathbf{c}) \models \bigvee_{j=1}^{m_a} \psi_j^a$ and is ***disabled*** otherwise. So an agent is enabled if it can perform a call. An agent a is ***selected*** in $\mathbf{c}$ if it is the caller in the call that for some $\mathbf{c}'$ determines the transition $\mathbf{c} \to \mathbf{c}'$ in ξ. Finally, a computation ξ is called a ***fair computation*** if it is finite or each agent that is enabled in infinitely many sequences in ξ is selected in infinitely many sequences in ξ.

We note in passing that various alternative definitions of fairness are possible; we just focus on one of them. An interested reader may consult [2], where several fairness definitions (for instance one focusing on actions and not on agents) for distributed programs were considered and compared.

We conclude this section by observing the following. Our definition of computation tree for protocol P presupposes that guards ψ_j^a are interpreted over the gossip model $\mathcal{M} = (\mathbf{C}^{<\omega}, \{\sim_a\}_{a \in \mathsf{A}})$. This means that when evaluating guards, agents consider as possible call sequences that cannot be generated by P. In other words, agents do not know the protocol. To model common knowledge of the considered protocol in the gossip model one should take as the domain of the gossip model $\mathcal{M}$ the underlying computation tree. However, the computation tree is defined by means of the underlying gossip model. To handle such a circularity an appropriate fixpoint definition is needed. We leave this topic for future work.

2.3 Correctness

[3]Note that while the sequences that are elements of the computation tree of a protocol are always finite (although possibly infinite in number), computations can be infinite sequences (of finite call sequences).

We are interested in proving the correctness of gossip protocols. Assume a gossip protocol P that is a parallel composition of the component programs $*[[]_{j=1}^{m_a} \psi_j^a \to \mathsf{c}_j^a]$.

We say that P is ***partially correct***, in short ***correct***, if in all situations sequences $\mathbf{c}$ that are leaves of the computation tree of P, for each agent a

$$(\mathcal{M}, \mathbf{c}) \models \bigwedge_{b \in \mathsf{A}} F_a B,$$

i.e., if for all situations sequences $\mathbf{c}$ that are leaves of the computation tree of P, each agent is an expert in the gossip situation $\mathbf{c}(\mathsf{root})$.

We say furthermore that P ***terminates*** if all its computations are finite and that P ***fairly terminates*** if all its fair computations are finite.

In the next section we provide examples showing that partial correctness and termination of the considered protocols can depend on the assumed mode of communication and on the number of agents. In what follows we study various gossip protocols and their correctness. We begin with the following obvious observation.

FACT 2.7. *For each protocol P the following implications ($\Rightarrow$) hold, where $T_P(x)$ stands for its termination and $FT_P(x)$ for its fair termination in a communication mode x:*

$$T_P(x) \Rightarrow FT_P(x).$$

Protocol R3 given in Section 4 shows that none of these implications can be reversed. Moreover, it is not the case either that for each protocol P:

$$T_P(\triangleright) \Rightarrow T_P(\text{push-pull}),$$
$$T_P(\triangleleft) \Rightarrow T_P(\text{push-pull}).$$

EXAMPLE 2.8. *Let $\mathsf{A} = \{a, b, c\}$ and define the following expression:*

$$\mathcal{A} \subset \mathcal{C} \quad := \quad \bigwedge_{I \in \{A,B,C\}} (F_a I \to F_c I) \wedge \bigvee_{I \in \{A,B,C\}} (F_c I \wedge \neg F_a I)$$

Expression $\mathcal{B} \subset \mathcal{C}$ can be defined analogously. Note that we denote by I the secret of agent i. Intuitively, $\mathcal{A} \subset \mathcal{C}$ means that agent c is familiar with all the secrets that agent a is familiar with, but not vice versa. So c is familiar with a superset of the secrets a is aware of. Further, let Exp_j stand for $\bigwedge_{I \in \{A,B,C\}} F_j I$.

Consider now the following component programs:

- *for agent a: $*[K_a(\neg(\mathcal{A} \subset \mathcal{C}) \wedge \neg Exp_a) \to a \triangleright c]$,*
- *for agent b: $*[K_b(\neg(\mathcal{B} \subset \mathcal{C}) \wedge \neg Exp_b) \to b \triangleright c]$,*
- *for agent c: $*[[]_{i \in \{a,b\}} K_c Exp_c \wedge \neg K_c Exp_i \to c \triangleright i]$.*

First note that in our logic, $K_i(\phi_1 \wedge \phi_2)$ is equivalent to $K_i\phi_1 \wedge K_i\phi_2$.

This protocol is correct. Indeed, initially, it is not the case that c knows to be an expert, hence the guard of c is false. Likewise, the guards of a and b are true; a for instance knows that c is not familiar with more secrets than a, and that a is not familiar with all secrets. So initially both a or b are enabled. If the first call is granted to a, this agent will call c, yielding the situation $A.B.AC$. Note that now, the guard of a is false (from a's perspective, c may now well be familiar with all secrets), the guard of b is true, and the guard of c is still false. So now only b is enabled, which yields the situation $A.B.ABC$. At this stage, only agent c is enabled and after he calls both a and b all guards become false.

Moreover, this protocol terminates. Indeed, the only computations are the ones in which first the calls $a \triangleright c$ and $b \triangleright c$ take place, in any order, followed by the calls $c \triangleright a$ and $c \triangleright b$, also performed in any order. However, if we use the push-pull mode instead of push, the call ac can be indefinitely repeated, so the protocol does not terminate. □

3. TWO SYMMETRIC PROTOCOLS

In this section we consider protocols for the case when the agents form a complete graph. We study two protocols. We present them first for the communication mode push-pull. (Partial) correctness of the considered protocols does not depend on the assumed mode of communication.

Learn new secrets protocol (LNS).

Consider the following program for agent i:

$$*[[]_{j \in \mathsf{A}} \neg F_i J \rightarrow (i,j)].$$

Informally, agent i calls agent j if i is not familiar with j's secret. Note that the guards of this protocol do not use the epistemic operator K_i, but they are equivalent to the ones that do, as $\neg F_i J$ is equivalent to $K_i \neg F_i J$.

This protocol was introduced in [3] and studied with respect to the push-pull mode, assuming asynchronous communication. As noted there this protocol is clearly correct. Also, it always terminates since after each call (i,j) the size of $\{(i,j) \in \mathsf{A} \times \mathsf{A} \mid \neg F_i J\}$ decreases. The same argument shows termination if the communication mode is pull.

However, if the communication mode is push, the protocol may fail to terminate, even fairly. To see it fix an agent a and consider a sequence of calls in which each agent calls a. At the end of this sequence a becomes an expert but nobody is familiar with his secret. So any extension of this sequence is an infinite computation.

Let us consider now the possible call sequences generated by the computations of this protocol. Assume that there are $n \geq 4$ agents. By the result mentioned in the introduction in each terminating computation at least $2n-4$ calls are made.

The LNS protocol can generate such shortest sequences (among others). Indeed, let $\mathsf{A} = \{a, b, c, d, i_1, \ldots, i_{n-4}\}$ be the set of agents. Then the following sequence of $2n-4$ calls

$$\begin{array}{c} (a,i_1),(a,i_2),\ldots,(a,i_{n-4}), \\ (a,b),(c,d),(a,c),(b,d), \\ (i_1,b),(i_2,b),\ldots,(i_{n-4},b) \end{array} \tag{1}$$

corresponds to a terminating computation.

The guards used in this protocol entail that after a call (i,j) neither the call (j,i) nor another call (i,j) can take place, that is between each pair of agents at most one call can take place. Consequently, the longest possible sequence contains at most $\frac{n(n-1)}{2}$ calls. Such a worst case can be generated by means of the following sequence of calls:

$$[2],\ [3],\ [4],\ldots,[n],$$

where for a natural number k, $[k]$ stands for the sequence $(1,k)$, $(2,k)$, …, $(k-1,k)$.[4]

Hear my secret protocol (HMS).

Next, we consider a protocol with the following program for agent i:

$$*[[]_{j \in \mathsf{A}} \neg K_i F_j I \rightarrow (i,j)].$$

[4]Other longest sequences are obviously possible, for instance: $12, 13, ..., 1n, 23, 24, ..., 2n, 34, 35, .., 3n, ..., (n-1)n$.

Informally, agent i calls agent j if he (agent i) does not know whether j is familiar with his secret. To prove correctness of this protocol it suffices to note that its exit condition

$$\bigwedge_{i,j \in \mathsf{A}} K_i F_j I$$

implies $\bigwedge_{i,j \in \mathsf{A}} F_j I$. To prove termination it suffices to note that after each call (i,j) the size of the set $\{(i,j) \mid \neg K_i F_j I\}$ decreases.

If the communication mode is push, then the termination argument remains valid, since after the call $i \triangleright j$ agent j still learns all the secrets agent i is familiar with.

However, if the communication mode is pull, then the protocol may fail to terminate, even fairly. To see it fix an agent j and consider the calls $i \triangleleft j$, where i ranges over $\mathsf{A} \setminus \{j\}$, arbitrarily ordered. Denote this sequence by $\mathbf{c}$. Consider now an infinite sequence of calls resulting from repeating $\mathbf{c}$ indefinitely. It is straightforward to check that such a sequence corresponds to a possible computation. Indeed, in this sequence agent j never calls and hence never learns any new secret. So for each $i \neq j$ the formula $\neg K_i F_j I$ remains true and hence each agent $i \neq j$ remains enabled. Moreover, after the calls from $\mathbf{c}$ took place agent j is not anymore enabled. Hence the resulting infinite computation is fair.

When there are $n \geq 4$ agents, the extreme cases in terms of the lengths of possible call sequences are the same as in the case of the LNS protocol. Indeed, let $\mathsf{A} = \{a, b, c, d, i_1, \ldots, i_{n-4}\}$ be the set of agents. Then the sequence of (1) corresponds to a terminating computation. Further, this protocol can generate computations in which $\frac{n(n-1)}{2}$ calls are made. The argument is the same as for the LNS protocol.

4. PROTOCOLS OVER DIRECTED RINGS

In this section we consider the case when the agents are arranged in a directed ring, where $n \geq 3$. For convenience we take the set of agents to be $\{1, 2, \ldots, n\}$. For $i \in \{1, \ldots, n\}$, let $i \oplus 1$ and $i \ominus 1$ denote respectively the successor and predecessor of agent i. That is, for $i \in \{1, \ldots, n-1\}$, $i \oplus 1 = i+1$, $n \oplus 1 = 1$, for $i \in \{2, \ldots, n\}$, $i \ominus 1 = i-1$, and $1 \ominus 1 = n$. For $k > 1$ we define $i \oplus k$ and $i \ominus k$ by induction in the expected way. Again, when reasoning about the protocols we denote the secret of agent $i \in \{1, \ldots, n\}$ by I. We consider four different protocols and study them with respect to their correctness and (fair) termination.

In this set up, a call sequence over a directed ring is a (possibly infinite) sequence of calls, all being of the same communication mode, and all involving an agent i and $i \oplus 1$. As before, we use $\mathbf{c}$ to denote such a call sequence and $\mathbf{C}_{DR}$ to denote the set of all call sequences over a directed ring. In this section, unless stated otherwise, by a call sequence we mean a sequence over a directed ring. The set of all such finite call sequences is denoted $\mathbf{C}_{DR}^{<\omega}$. A gossip model for a directed ring is a tuple $\mathcal{M}_{DR} = (\mathbf{C}_{DR}^{<\omega}, \{\sim_a\}_{a \in \mathsf{A}})$, where each $\sim_a \subseteq \mathbf{C}_{DR}^{<\omega} \times \mathbf{C}_{DR}^{<\omega}$ is as in Definition 2.4. The truth definition is as before, and the notion of a ***computation tree for directed rings*** $\mathbf{C}_{DR}^{P} \subseteq \mathbf{C}_{DR}^{<\omega}$ of a ring protocol P is analogous to the notion defined before. Note that by restricting the domain in $\mathcal{M}_{DR}$ to $\mathbf{C}_{DR}^{<\omega}$, the ring network—and hence who is the successor of whom—becomes common knowledge.

When presenting the protocols we use the fact that $F_i J$ is equivalent to $K_i F_i J$.

Ring protocol R1.

Consider first a gossip protocol with the following program for i:

$$*[\bigvee_{j=1}^{n}(F_iJ \wedge K_i\neg F_{i\oplus 1}J) \rightarrow i\Diamond i\oplus 1],$$

where $\Diamond$ denotes the mode of communication, so $\triangleright$, $\triangleleft$ or push-pull.

Informally, agent i calls his successor, agent $i\oplus 1$, if i is familiar with some secret and he knows that his successor is not familiar with it.

PROPOSITION 4.1. *Let $\Diamond = \triangleright$. Protocol R1 terminates and is correct.*

Termination and correctness do not both hold for the other communication modes. Consider first the pull communication mode, i.e., $\Diamond = \triangleleft$. Then the protocol does not always terminate. Indeed, each call $i \triangleleft i \oplus 1$ can be repeated. Next, consider the push-pull communication mode. We show that then the protocol is not correct. Indeed, take

$$\mathbf{c} = (1,2),\ (2,3), \ldots, (n-1,n).$$

We claim that after the sequence of calls $\mathbf{c}$ the exit condition of the protocol is true. To this end we consider each agent in turn.

After $\mathbf{c}$ each agent i, where $i \neq n$ is familiar the secrets of the agents $1, 2, \ldots, i+1$. Moreover, because of the call $(i, i+1)$ agent i knows that agent $i+1$ is familiar with these secrets. So the exit condition of agent i is true.

To deal with agent n note that $\mathbf{c} \sim_n \mathbf{c}.(n-2, n-1).(n-3, n-2).\ldots.(2,3).(1,2)$. After the latter call sequence agent 1 becomes an expert. So after $\mathbf{c}$ agent n cannot know that agent 1 is not familiar with some secret. Consequently, after $\mathbf{c}$ the exit condition of agent n is true, as well. However, after $\mathbf{c}$ agent 1 is not an expert, so the protocol is indeed not correct.

In what follows we initially present the protocols assuming the push-pull mode of communication.

Ring protocol R2.

Consider now a gossip protocol with the following program for agent i:

$$*[\neg K_iF_{i\oplus 1}I \ominus 1 \rightarrow (i, i\oplus 1)],$$

where (recall) $I \ominus 1$ denotes the secret of agent $i \ominus 1$. Informally, agent i calls his successor, agent $i \oplus 1$, if i does not know that his successor is familiar with the secret of i's predecessor, i.e., agent $i \ominus 1$.

PROPOSITION 4.2. *If $|\mathsf{A}| \in \{3,4\}$ then protocol R2 is correct.*

However, this protocol is not correct for five or more agents. To see it consider the sequence of calls

$$(1,2),\ (2,3),\ \ldots, (n-1,n),\ (n,1),\ (1,2)$$

where $n \geq 5$. After it the exit condition of the protocol is true. However, agent 3 is not familiar with the secret of agent 5.

Note that the same argument shows that the protocol in which we use $\neg K_iF_{i\oplus 1}I \vee \neg K_iF_{i\oplus 1}I \ominus 1$ instead of $\neg K_iF_{i\oplus 1}I \ominus 1$ is incorrect, as well.

Moreover, this protocol does not always terminate. Indeed, one possible computation consists of an agent i repeatedly calling his successor $i \oplus 1$.

Protocol	T	FT	T for $\triangleright$	FT for $\triangleright$	T for $\triangleleft$	FT for $\triangleleft$
LNS	yes	yes	no	no	yes	yes
HMS	yes	yes	yes	yes	no	no
R3	no	yes	no	yes	no	yes
R4	yes	yes	yes	yes	no	yes

Table 1: Summary of termination results.

Ring protocol R3.

Next, consider the following modification of protocol R2 in which we use the following program for agent i:

$$*[(\neg \bigwedge_{j=1}^{n} F_iJ) \vee \neg K_iF_{i\oplus 1}I \ominus 1 \rightarrow (i, i\oplus 1)].$$

Informally, agent i calls his successor, agent $i\oplus 1$, if i is not familiar with all the secrets or i does not know that his successor is familiar with the secret of his predecessor, agent $i \ominus 1$.

This gossip protocol is obviously correct thanks to the fact that $\bigwedge_{i=1}^{n}\bigwedge_{j=1}^{n} F_iJ$ is part of the exit condition. However, it does not always terminate for the same reason as the previous one.

On the other hand, the following holds.

PROPOSITION 4.3. *Protocol R3 fairly terminates.*

The same conclusions concerning non termination and fair termination can be drawn for the push and the pull modes of communication. Indeed, for push it suffices to consider the sequence of calls $i \triangleright i\oplus 1,\ i\oplus 1 \triangleright i\oplus 2, \ldots, i\ominus 1 \triangleright i$ after which agent $i \ominus 1$ becomes disabled, and for pull the sequence of calls $i \triangleleft i\oplus 1,\ i\ominus 1 \triangleleft i, \ldots, i\oplus 2 \triangleleft i\oplus 3$ after which agent $i \oplus 2$ becomes disabled.

Ring protocol R4.

Finally, we consider a protocol that is both correct and terminates for the push-pull mode. Consider the following program for i:

$$*[\bigvee_{j=1}^{n}(F_iJ \wedge \neg K_iF_{i\oplus 1}J) \rightarrow (i, i\oplus 1)].$$

Informally, agent i calls his successor, agent $i \oplus 1$, if i is familiar with some secret and he does not know whether his successor is familiar with it. Note the similarity with protocol R1.

PROPOSITION 4.4. *Protocol R4 terminates and is correct.*

If the communication mode is push, then the termination argument remains valid, since after the call $i \triangleright i\oplus 1$ agent $i\oplus 1$ still learns all the secrets that agent i is familiar with and hence the above set $\{(i,j) \mid \neg K_iF_{i\oplus 1}J\}$ decreases.

If the communication mode is pull, then the protocol may fail to terminate, because after the first call $i \triangleleft i \oplus 1$ agent $i \oplus 1$ does not learn the secret of agent i and consequently the call can be repeated. However, the situation changes when fairness is assumed.

PROPOSITION 4.5. *For the pull communication mode protocol R4 fairly terminates.*

Table 1 summarizes the termination properties of the protocols considered in the paper.

5. CONCLUSIONS

The aim of this paper was to introduce distributed gossip protocols, to set up a formal framework to reason about them, and to illustrate it by means of an analysis of selected protocols.

Our results open up several avenues for further research. First, our correctness arguments were given in plain English with occasional references to epistemic tautologies, such as $K_i\phi \rightarrow \phi$, but it should be possible to formalize them in a customized epistemic logic. Such a logic should have a protocol independent component that would consist of the customary S5 axioms and a protocol dependent component that would provide axioms that depend on the mode of communication and the protocol in question. An example of such an axiom is the formula $K_i F_{i\oplus 1} I \ominus 1 \rightarrow F_i I \oplus 1$ that we used when reasoning about protocol R2. To prove the validity of the latter axioms one would need to develop a proof system that allows us to compute the effect of the calls, much like the computation of the strongest postconditions in Hoare logics. Once such a logic is provided the next step will be to study formally its properties, including decidability. Then we could clarify whether the provided correctness proofs could be carried out automatically.

Second, generalizing further the ideas we introduced by considering directed rings, gossip protocols could be studied in interface with network theory (see [13] for a textbook presentation). Calls can be assumed to be constrained by a network, much like in the literature on 'centralized' gossip (cf. [10]) or even have probabilistic results (i.e., secrets are passed with given probabilities). More complex properties of gossip protocols could then be studied involving higher-order knowledge or forms of group knowledge among neighbors (e.g., "it is common knowledge among a and her neighbors that they are all experts"), or their stochastic behavior (e.g., "at some point in the future all agents are experts with probability p").

Third, it will be interesting to analyze the protocols for the types of calls considered in [3]. They presuppose some form of knowledge that a call took place (for instance that given a call between a and b each agent $c \neq a, b$ noted the call but did not learn its content). Another option is to consider multicasting (calling several agents at the same time).

Finally, many assumptions of the current setup could be lifted. Different initial and final situations could be considered, for instance common knowledge of protocols could be assumed, or common knowledge of the familiarity of all agents with all the secrets upon termination could be required. Finally, to make the protocols more efficient passing of tokens could be allowed instead of just the transmission of secrets by means of calls.

Acknowledgments

We would like to thank Hans van Ditmarsch and the referees for helpful comments and Rahim Ramezanian for useful comments about Example 2.8. This work resulted from a research visit by Krzysztof Apt to Davide Grossi and Wiebe van der Hoek, sponsored by the 2014 Visiting Fellowship Scheme of the Department of Computer Science of the University of Liverpool. The first author is also a Visiting Professor at the University of Warsaw. He was partially supported by the NCN grant nr 2014/13/B/ST6/01807.

6. REFERENCES

[1] K. R. Apt, F. R. de Boer, and E. R. Olderog. *Verification of Sequential and Concurrent Programs*. Springer, 2009.

[2] K. R. Apt, N. Francez, and S. Katz. Appraising fairness in distributed languages. *Distributed Computing*, 2(4):226–241, 1988.

[3] M. Attamah, H. van Ditmarsch, D. Grossi, and W. Van der Hoek. Knowledge and gossip. In *Proceedings of ECAI'14*, pages 21–26. IOS Press, 2014.

[4] B. Baker and R. Shostak. Gossips and telephones. *Discrete Mathematics*, 2:197–193, 1972.

[5] R. Bumby. A problem with telephones. *SIAM Journal of Algorithms and Discrete Methods*, 2:13–18, 1981.

[6] E. W. Dijkstra. Guarded commands, nondeterminacy and formal derivation of programs. *Communications of the ACM*, 18:453–457, 1975.

[7] R. Fagin, J. Halpern, Y. Moses, and M. Vardi. Knowledge-based programs. *Distributed Computing*, 10:199–225, 1997.

[8] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning about knowledge*. The MIT Press, Cambridge, 1995.

[9] A. Hajnal, E. C. Milner, and E. Szemeredi. A cure for the telephone disease. *Canadian Mathematical Bulletin*, 15:447–450, 1972.

[10] S. M. Hedetniemi, S. T. Hedetniemi, and A. L. Liestman. A survey of gossiping and broadcasting in communication networks. *Networks*, 18(4):319–349, 1988.

[11] C. A. R. Hoare. Communicating sequential processes. *Communications of the ACM*, 21:666–677, 1978.

[12] INMOS Limited. *Occam Programming Manual*. Prentice-Hall International, 1984.

[13] M. O. Jackson. *Social and Economic Networks.* Princeton University Press, 2008.

[14] R. Kurki-Suonio. Towards programming with knowledge expressions. In *Proceedings of POPL'86*, pages 140–149, 1986.

[15] J. Meyer and W. van der Hoek. *Epistemic Logic for AI and Computer Science*, volume 41 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1995.

[16] R. Parikh and R. Ramanujam. Distributed processing and the logic of knowledge. In *Logic of Programs*, LNCS 193, pages 256–268. Springer, 1985. Similar to *JoLLI* 12: 453–467, 2003.

[17] A. Seress. Quick gossiping without duplicate transmissions. *Graphs and Combinatorics*, 2:363–383, 1986.

[18] R. Tijdeman. On a telephone problem. *Nieuw Archief voor Wiskunde*, 3(XIX):188–192, 1971.

APPENDIX

PROOF OF PROPOSITION 4.1.

Termination Given a call sequence $\mathbf{c}$ define the set

$$Inf(\mathbf{c}) := \{(i,j) \mid i,j \in \{1,\ldots,n\} \text{ and } (\mathcal{M}_{DR}, \mathbf{c}) \models F_i J\}.$$

After each enabled call $i \triangleright i \oplus 1$ in $\mathbf{c}$, the set $Inf(\mathbf{c})$ increases, which ensures termination since each set $Inf(\cdot)$ has at most n^2 elements.

Correctness Consider a leaf of the computation tree. Then the exit condition

$$\bigwedge_{i=1}^{n} \bigwedge_{j=1}^{n} (\neg F_i J \vee \neg K_i \neg F_{i\oplus 1} J)$$

is true. We proceed by induction to show that then each $F_i J$ is true, where $i,j \in \{1,\ldots,n\}$, and where the pairs (i,j) are ordered as follows:

$$\begin{aligned}&(1,1),(2,1),\ldots,(n,1),\\ &(2,2),(3,2),\ldots,(1,2),\\ &\ldots,\\ &(n,n),(1,n),\ldots,(n-1,n).\end{aligned}$$

So the ith row lists the pairs (j,i) with $j \in \{1,\ldots,n\}$ ranging clockwise, starting at i.

Take a pair (i,j). If $i=j$, then F_iJ is true by assumption. If $i \neq j$, then consider the pair that precedes it in the above ordering. It is then of the form (i_1,j), where $i = i_1 \oplus 1$. By the induction hypothesis $F_{i_1}J$ is true, so by the exit condition $\neg K_{i_1}\neg F_iJ$ is true.

Suppose now towards a contradiction that $\neg F_{i_1\oplus 1}J$ is true. Then $i_1 \oplus 1 \neq j$. Hence by virtue of the considered communication mode and Definition 2.4 it follows that agent i_i knows that $\neg F_{i_1\oplus 1}J$ is true since the only way for $i_1 \oplus 1$ to become familiar with J is by means of a call from i_1. So $K_{i_1}\neg F_iJ$ is true. This yields a contradiction. Hence F_iJ is true.

So we showed, as desired, that $\bigwedge_{i=1}^n \bigwedge_{j=1}^n F_iJ$ is true in the considered leaf. $\square$

PROOF OF PROPOSITION 4.2. To start with, $\bigwedge_{i=1}^n F_iI$ is true in every node of the computation tree. Suppose the exit condition $\bigwedge_{i=1}^n K_iF_{i\oplus 1}I \ominus 1$ is true at a node of the computation tree (in short, true). It implies that $\bigwedge_{i=1}^n F_{i\oplus 1}I \ominus 1$ is true. Fix $i \in \{1,\ldots,n\}$. By the above $F_iI \ominus 2$ is true. Further, the implication $K_iF_{i\oplus 1}I \ominus 1 \rightarrow F_iI \ominus 1$ is true in every node of the computation tree (remember, the agents are positioned on a directed ring). If $n=3$, this proves that $\bigwedge_{j=1}^n F_iJ$ is true.
If $n=4$, we note that $K_iF_{i\oplus 1}I \ominus 1$ implies that agent $i \oplus 1$ learned $I \ominus 1$ through a call of agent i and hence the implication $K_iF_{i\oplus 1}I \ominus 1 \rightarrow F_iI \oplus 1$ is true in every node of the computation tree, as well (remember that the mode is push-pull). We conclude that $\bigwedge_{j=1}^n F_iJ$ is true. $\square$

PROOF OF PROPOSITION 4.3. First, note that the following three statements are equivalent for each node $\mathbf{c}$ of an arbitrary computation ξ and each agent i:

- i is disabled at $\mathbf{c}$,
- $(\mathcal{M}_{DR},\mathbf{c}) \models (\bigwedge_{j=1}^n F_iJ) \wedge K_iF_{i\oplus 1}I \ominus 1$,
- a sequence of calls $(i\oplus 2, i\oplus 3),\ (i\oplus 3, i\oplus 4),\ldots,(i,i\oplus 1)$ (possibly interspersed with other calls) has taken place in ξ before $\mathbf{c}$.

Suppose now towards a contradiction that an infinite fair computation ξ exists. We proceed by case distinction.

Case 1 Some agent becomes disabled in ξ.
We claim that if an agent i becomes disabled in ξ, then also agent $i \oplus 1$ becomes disabled in ξ. Indeed, otherwise by fairness at some point in ξ after which i becomes disabled, agent $i \oplus 1$ calls his successor, $i \oplus 2$, and by the above sequence of equivalences in turn becomes disabled.

We conclude by induction that at some point in ξ all agents become disabled and hence ξ terminates, which yields a contradiction.

Case 2 No agent becomes disabled in ξ.
By fairness each agent calls in ξ infinitely often his successor. So for every agent i there exists in ξ the sequence of calls $(i\oplus 2, i\oplus 3),\ (i\oplus 3, i\oplus 4),\ldots,(i,i\oplus 1)$ (possibly interspersed with other calls). By the above sequence of equivalences after this sequence of calls agent i becomes disabled, which yields a contradiction. $\square$

PROOF OF PROPOSITION 4.4.
Termination It suffices to note that after each call $(i,i\oplus 1)$ the size of the set

$$\{(i,j) \in \mathsf{A}\times\mathsf{A} \mid \neg K_iF_{i\oplus 1}J\}$$

decreases.
Correctness Consider a leaf of the computation tree. Then the exit condition

$$\bigwedge_{i=1}^n \bigwedge_{j=1}^n (\neg F_iJ \vee K_iF_{i\oplus 1}J)$$

is true. As in the case of protocol R1 we prove that it implies each F_iJ is true by induction on the pairs (i,j), where $i,j \in \{1,\ldots,n\}$, ordered as follows:

$$\begin{aligned}&(1,1),(2,1),\ldots,(n,1),\\ &(2,2),(3,2),\ldots,(1,2),\\ &\ldots,\\ &(n,n),(1,n),\ldots,(n-1,n).\end{aligned}$$

Take a pair (i,j). If $i=j$, then F_iJ is true by assumption. If $i \neq j$, then consider the pair that precedes it in the above ordering, so (i_1,j), where $i = i_1 \oplus 1$. By the induction hypothesis $F_{i_1}J$ is true, so by the exit condition $K_{i_1}F_iJ$ is true and hence F_iJ is true. $\square$

PROOF OF PROPOSITION 4.5. Consider the following sequence of statements:

(i) i is disabled at $\mathbf{c}$,
(ii) $(\mathcal{M}_{DR},\mathbf{c}) \models \bigwedge_{j=1}^n (F_iJ \rightarrow K_iF_{i\oplus 1}J)$,
(iii) $(\mathcal{M}_{DR},\mathbf{c}) \models K_iF_{i\oplus 1}$,
(iv) a sequence of calls $i\ominus 1 \triangleleft i, i\ominus 2 \triangleleft i\ominus 1,\ldots,i\triangleleft i\oplus 1$ (possibly interspersed with other calls) has taken place in ξ before $\mathbf{c}$.

It is easy to verify that these statements are logically related in the following way:

$$\text{(i)} \Leftrightarrow \text{(ii)} \Rightarrow \text{(iii)} \Rightarrow \text{(iv)} \Rightarrow \text{(ii)}$$

for each node $\mathbf{c}$ of an arbitrary computation ξ and each agent i. They are therefore equivalent. Suppose now towards a contradiction that an infinite fair computation ξ exists. As in the proof of Proposition 4.3 we proceed by case distinction.

Case 1 Some agent becomes disabled in ξ.
We claim that if an agent i becomes disabled in ξ, then also $i \ominus 1$ becomes disabled in ξ. Indeed, otherwise by fairness at some point in ξ after which j becomes disabled, agent $i \ominus 1$ calls his successor, i, and by the above sequence of equivalences in turn becomes disabled.

We conclude by induction that at some point in ξ all agents become disabled and hence ξ terminates, which yields a contradiction.

Case 2 No agent becomes disabled in ξ.
By fairness each agent calls in ξ infinitely often his successor. So for every agent i there exists in ξ a sequence of calls $i\ominus 1 \triangleleft i, i\ominus 2 \triangleleft i\ominus 1,\ldots,i\triangleleft i\oplus 1$ (possibly interspersed with other calls). By the above sequence of equivalences, after this sequence of calls agent i becomes disabled, which yields a contradiction. $\square$