Propositional Gossip Protocols

Joseph Livesey and Dominik Wojtczak

University of Liverpool, UK
{joseph.livesey,d.wojtczak}@liverpool.ac.uk

Abstract. Gossip protocols are programs that can be used by a group of n agents to synchronise what they know. Namely, assuming each agent holds a secret, the goal of a protocol is to reach a situation in which all agents know all secrets. Distributed epistemic gossip protocols use epistemic formulas in the component programs for the agents. In this paper, we solve open problems regarding one of the simplest classes of such gossip protocols: propositional gossip protocols, in which whether an agent can make a call depends only on his currently known set of secrets. Specifically, we show that all correct propositional gossip protocols, i.e., the ones that always terminate in a situation where all agents know all secrets, require the underlying undirected communication graph to be complete and at least $2n - 3$ calls to be made. We also show that checking correctness of a given propositional gossip protocol is a co-NP-complete problem. Finally we report on implementing such a check with model checker nuXmv.

1 Introduction

Gossip protocols have the goal of spreading information through a network via point-to-point communications (which we refer to as calls). Each agent holds initially a secret and the aim is to arrive at a situation in which all agents know each other secrets. During each call the caller and callee exchange all secrets that they know at that point. Such protocols were successfully used in a number of domains, for instance communication networks [19], computation of aggregate information [23], and data replication [25]. For a more recent account see [22] and [24]. One of the early results established by a number of authors in the seventies, e.g.,[26], is that for n agents $2n - 4$ calls are necessary and sufficient when every agent can communicate with any other agent. When such a communication graph is not complete, $2n - 3$ calls may be needed [12] but are sufficient for any connected communication graph [18]. However, all such protocols considered in these papers were centralised.

In [10] a dynamic epistemic logic was introduced in which gossip protocols could be expressed as formulas. These protocols rely on agents' knowledge and are distributed, so they are *distributed epistemic gossip protocols*. This also means that they can be seen as special cases of knowledge-based programs introduced in [16].

In [2] a simpler modal logic was introduced that is sufficient to define these protocols and to reason about their correctness. This logic is interesting in its own rights and was subsequently studied in a number of papers. In this paper, we are going to focus on its simplest propositional fragment.

Propositional gossip protocols are a particular type of epistemic gossip protocol in which all guards are propositional. This means that calls being made by each agent are

dependent only on the secrets that the agent have had access to. Clearly, this can lead to states where multiple calls are possible at the same time. Then a scheduler would decide which call takes priority. Throughout this paper, we assume that the scheduler is demonic and it picks the order of calls in a way such that the protocol fails or to maximise the number of calls made before termination. In other words, we study these gossip protocols in their worst-case scenario.

In [9], many challenging open problems about general as well as propositional gossip protocols were listed. In the paper we manage to resolve some of them. In particular, we solve its Problem 5, which asks for a characterisation of the class of graphs for which propositional protocols exist. In Section 3 we show that, when we ignore the direction of edges in the communication graph, the only class possible is complete graphs. In order to show that we need to establish many interesting properties of computations of such protocols. We also partially resolve its Problem 6, which asks to show that a gossip protocol needs at least $2n - 3$ calls to be correct. We prove this is indeed the case at least for the class of propositional gossip protocols. Note that, unlike what was shown in [12], this lower bound holds even if the communication graph is a complete graph. Finally, we also partially address Problem 7, which asks for the precise computational complexity of checking the correctness of such protocols. It is known that for gossip protocols without nesting of modalities this problem is in coNP^{NP} [4]. We improve this to co-NP-completeness for propositional ones.

Related work. Much work has been done on general epistemic gossip protocols. The various types of calls used in [10] and [2] were presented in a uniform framework in [3], where in total 18 types of communication were considered and compared w.r.t. their epistemic strength. In [5], and its full version [8], the decidability of the semantics of the gossiping logic and truth was established for its limited fragment (namely, without nesting of modalities). Building upon these results it was proved in [5] that the distributed gossip protocols, the guards of which are defined in this logic, are implementable, that their partial correctness is decidable, and in [7] that termination and two forms of fair termination of these protocols are decidable, as well. Building on that, [30] showed decidability of the full logic for various variants of the gossiping model. Further, in [4] the computational complexity of this fragment was studied and in [6] an extension with the common knowledge operator was considered and analogous decidability results were established there.

Despite how simple this modal logic seems to be, there remain natural open problems about it and the gossip protocols defined using it. These problems were discussed at length in [9], where partial results were also presented that be build upon in this paper. Some of these open problems were subsequently tackled in [30], but propositional protocols were not studied there and questions regarding them were left open.

Centralised gossip protocols were studied in [20] and [21]. These had the goal to achieve higher-order shared knowledge. This was investigated further in [14], where optimal protocols for various versions of such a generalised gossip problem were given. These protocols depend on various parameters, such as the type of the underlying graph or communication. Additionally, different gossip problems which contained some negative goals, for example that certain agents must not know certain secrets, were studied. Such

problems were further studied in [15] with temporal constraints, i.e., a given call has to (or can only) be made within a given time interval.

The number of calls needed to reach the desired all expert situation in the distributed but synchronous setting was studied in [27]. In the synchronous setting, agents are notified if a call was made, but may not necessary know which agents were involved. In this paper we study the more complex fully distributed asynchronous setting, where agents are not aware of the calls they do not participate in. In [29,28] the expected time of termination of several gossip protocols for complete graphs was studied.

Dynamic distributed gossip protocols were studied in [31], in which the calls allow the agents to transmit the links as well as share secrets. These protocols were characterised in terms of the class of graphs for which they terminate. Various dynamic gossip protocols were proposed and analysed in [32]. In [17] these protocols were analysed by embedding them in a network programming language NetKAT [1].

Structure of the paper. We first introduce the logic, originally defined in [2], and then move on to tackle some open problems for the propositional gossip protocols. The first aim is to look at the communication graph required for the existence of a correct propositional gossip protocol (Section 3) and then a lower bound on the number of calls needed by such a protocol (Section 4). We then move on to looking at the complexity of the natural decision problems for these protocols (Section 5), before touching on some computational attempts to see how quickly a computer can determine the correctness of a given propositional gossip protocol. Due to the space limit some details of the proofs are omitted and will be published in a journal version of this paper later.

2 Gossiping Logic

We recall here the framework of [2], which we restrict to the propositional setting. We assume a fixed set A of $n \geq 3$ ***agents*** and stipulate 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 S the set of all secrets.

The propositional language $\mathcal{L}_p$ is defined by the following grammar:

$$\phi ::= F_a S \mid \neg\phi \mid \phi \wedge \phi,$$

where $S \in \mathsf{S}$ and $a \in \mathsf{A}$. We will distinguish the following sublanguage $\mathcal{L}_p^a$, where $a \in \mathsf{A}$ is a fixed agents, which disallow all F_b operators for $b \neq a$.

So $F_a S$ is an atomic formula, which we read as 'agent a is familiar with the secret S'. Note that in [2], a compound formula $K_a\phi$, i.e., 'agent a knows the formula ϕ is true', was used. Dropping $K_a\phi$ from the logic simplifies greatly its semantics and the execution of a gossip protocol, while it is still capable of describing a rich class of protocols. Below we shall freely use other Boolean connectives that can be defined using $\neg$ and $\wedge$ in a standard way. We shall use the following formula

$$Exp_i \equiv \bigwedge_{S \in \mathsf{S}} F_i S,$$

that denotes the fact that agent i is an ***expert***, i.e., he is familiar with all the secrets.

Each ***call***, written as ab or a, b, concerns two different agents, the ***caller***, a, and the ***callee***, b. After the call the caller and the callee learn each others secrets. 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. We refer to any such call an a***-call*** (b-call for agent b, etc.).

In what follows we focus on call sequences. Unless explicitly stated each call sequence is assumed to be finite. 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 finite call sequences. Given call sequences $\mathbf{c}$ and $\mathbf{d}$ and a call c we denote by $\mathbf{c}.\mathsf{c}$ the outcome of adding c at the end of the sequence $\mathbf{c}$ and by $\mathbf{c}.\mathbf{d}$ the outcome of appending the sequences $\mathbf{c}$ and $\mathbf{d}$. We say that $\mathbf{c}'$ is an ***extension*** of a call sequence $\mathbf{c}$ if for some call sequence $\mathbf{d}$ we have $\mathbf{c}' = \mathbf{c}.\mathbf{d}$.

To describe what secrets the agents are familiar with, we use the concept of a ***gossip situation***. It is a sequence $\mathsf{s} = (\mathsf{Q}_a)_{a \in \mathsf{A}}$, where $\{A\} \subseteq \mathsf{Q}_a \subseteq \mathsf{S}$ for each agent a. Intuitively, Q_a is the set of secrets a is familiar with in the gossip situation s. The ***initial gossip situation*** is the one in which each Q_a equals $\{A\}$ and is denoted by root. It reflects the fact that initially each agent is familiar only with his own secret. Note that an agent a is an expert in a gossip situation s iff $\mathsf{Q}_a = \mathsf{S}$.

Each call transforms the current gossip situation by modifying the sets of secrets the agents involved in the call are familiar with as follows. Consider a gossip situation $\mathsf{s} := (\mathsf{Q}_d)_{d \in \mathsf{A}}$ and a call ab.

Then

$$ab(\mathsf{s}) := (Q'_d)_{d \in \mathsf{A}},$$

where $\mathsf{Q}'_a = \mathsf{Q}'_b = \mathsf{Q}_a \cup \mathsf{Q}_b$, and for $c \notin \{a, b\}$, $\mathsf{Q}'_c = \mathsf{Q}_c$.

So the effect of a call is that the caller and the callee share the secrets they are familiar with.

The result of applying a call sequence to a gossip situation s is defined inductively as follows:

$$\epsilon(\mathsf{s}) := \mathsf{s},\ (\mathsf{c}.\mathbf{c})(\mathsf{s}) := \mathbf{c}(\mathsf{c}(\mathsf{s})).$$

Example 1. 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 a comma. e.g., if there are three agents, a, b and c, then $\mathsf{root} = A, B, C$ and the gossip situation $(\{A, B\}, \{A, B\}, \{C\})$ will be written as AB, AB, C.

Let $\mathsf{A} = \{a, b, c\}$. Consider the call sequence $ac.cb.ac$. It generates the following successive gossip situations starting from root:

$$A, B, C \xrightarrow{ac} AC, B, AC \xrightarrow{cb} AC, ABC, ABC \xrightarrow{ac} ABC, ABC, ABC.$$

Hence $(ac.cb.ac)(\mathsf{root}) = (ABC, ABC, ABC)$. □

Definition 2. *Consider a call sequence $\mathbf{c} \in \mathbf{C}$. We define the satisfaction relation $\models$ inductively as follows:*

$$\begin{aligned} \mathbf{c} &\models F_a S \textit{ iff } S \in \mathbf{c}(\mathsf{root})_a, \\ \mathbf{c} &\models \neg\phi \textit{ iff } \mathbf{c} \not\models \phi, \\ \mathbf{c} &\models \phi_1 \wedge \phi_2 \textit{ iff } \mathbf{c} \models \phi_1 \textit{ and } \mathbf{c} \models \phi_2. \end{aligned}$$

So a formula F_aS is true after the call sequence $\mathbf{c}$ whenever secret S belongs to the set of secrets agent a is familiar with in the situation generated by the call sequence $\mathbf{c}$ applied to the initial situation root. Hence $\mathbf{c} \models Exp_a$ iff agent a is an expert in $\mathbf{c}(\mathsf{root})$.

By a ***propositional 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 $m \geq 0$ and each $\psi_j \to \mathsf{c}_j$ is such that

- a is the caller in the call c_j,
- $\psi_j \in \mathcal{L}_p^a$.

We call each such construct $\psi \to \mathsf{c}$ a ***rule*** and refer in this context to ψ as a ***guard***.

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

We assume that in each gossip protocol the agents are the nodes of a directed graph (digraph) and that each call ab is allowed only if $a \to b$ is an edge in this digraph. A minimal digraph that satisfies this assumption is uniquely determined by the syntax of the protocol and we call this digraph the ***communication graph*** of a given protocol. Given that the aim of each gossip protocol is that all agents become experts it is natural to consider connected communication graphs only. On the other hand, the ***underlying undirected communication graph*** of a given protocol is the undirected graph we obtain when all directed edges in the communication graph are replaced with undirected ones connecting the same nodes.

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

The ***computation tree*** of P is a directed tree defined inductively as follows. Its nodes are call sequences and its root is the empty call sequence ϵ. Further, if $\mathbf{c}$ is a node and for some rule $\psi_j^a \to \mathsf{c}_j^a$ we have $\mathbf{c} \models \psi_j^a$, then $\mathbf{c}.\mathsf{c}_j^a$ is a node that is a direct descendant of $\mathbf{c}$. Intuitively, the arc from $\mathbf{c}$ to $\mathbf{c}.\mathsf{c}_j^a$ records the effect of the execution of the rule $\psi_j^a \to \mathsf{c}_j^a$ performed after the call sequence $\mathbf{c}$ took place.

By a ***computation*** of a gossip protocol P we mean a maximal rooted path in its computation tree. In what follows we identify each computation with the unique call sequence it generates. Any prefix of such a call sequence is called a ***prefix of*** P. We say that the gossip protocol P is ***partially correct*** if for all leaves $\mathbf{c}$ of the computation tree of P, and all agents a, we have $\mathbf{c} \models Exp_a$, i.e., if 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 say that P ***is correct*** if it is partially correct and terminates.

In [10] the following correct propositional gossip protocol, called *Learn New Secrets* (LNS in short), for complete digraphs was proposed.

Example 3 (LNS protocol). The following program is used by agent i:

$$*[[]_{j \in \mathsf{A}} \neg F_i J \to ij].$$

Informally, agent i calls agent j if agent i is not familiar with j's secret. □

We now define a new propositional protocol whose communication graph is not complete. First of all, agents will only be able to call agents with a higher "index", which for instance can be his phone number or name, with the corresponding total order ($>$) on A. Second, just like in the LNS protocol, agents can only call another agent if they do not know their secret. Finally, we require that an agent can make a call to another agent only if he already knows all the secrets of agents with the index value lower than the agent to be called. We will call this protocol ***Learn Next Secret (LXS)*** and its formal definition is as follows.

Example 4 (LXS protocol). The following program is used by agent i:

$$*[]_{\{j\in \mathsf{A} | j>i\}} \neg F_i J \wedge \bigwedge_{\{k\in \mathsf{A}|k<j\}} F_i K \rightarrow ij].$$

□

Note that although the communication graph of LXS protocol is not complete, its underlying undirected communication graph is, which we show is always the case for correct propositional protocols in the next section.

3 Required Communication Graph

We now show that for natural classes of connected graphs no correct propositional gossip protocol exists. We first show that by carefully removing some of the calls in a prefix of P one can get another prefix of P.

Lemma 5 (Call Removal). *Consider a propositional gossip protocol P. Let $\mathbf{c}.\mathbf{d}$ be a prefix of P such that $\mathbf{c}.\mathbf{d} \not\models F_a B$. Let $\mathbf{d}'$ be $\mathbf{d}$ where all calls that involve an agent familiar with B are removed, then $\mathbf{c}.\mathbf{d}'$ is also a prefix of P and, moreover, $(\mathbf{c}.\mathbf{d})(\mathsf{root})_a = (\mathbf{c}.\mathbf{d}')(\mathsf{root})_a$.*

Proof. It suffices to show that we can remove the last such call in $\mathbf{d}$, because that clearly preserves the $\mathbf{c}.\mathbf{d}' \not\models F_a B$ property and then we can simply repeat this procedure until no such calls are left in $\mathbf{d}$.

Let $\mathbf{d} = \mathbf{d}_1.cd.\mathsf{c}_1.\mathsf{c}_2 \ldots \mathsf{c}_k$, where cd is the last call that involves an agent that is familiar with B, i.e., $\mathbf{c}.\mathbf{d}_1 \models F_c B \vee F_d B$. Straight from the definition of the outcome of the cd call, for all agents $x \notin \{c, d\}$, $\mathbf{c}.\mathbf{d}_1(\mathsf{root})_x = (\mathbf{c}.\mathbf{d}_1.cd)(\mathsf{root})_x$. At the same time, agents c, d cannot be involved in any of the calls $\mathsf{c}_1, \ldots, \mathsf{c}_k$. Therefore, we also have for all agents $x \notin \{c, d\}$, $(\mathbf{c}.\mathbf{d}_1.\mathsf{c}_1)(\mathsf{root})_x = (\mathbf{c}.\mathbf{d}_1.cd.\mathsf{c}_1)(\mathsf{root})_x$, and by induction $(\mathbf{c}.\mathbf{d}_1.\mathsf{c}_1 \ldots \mathsf{c}_i)(\mathsf{root})_x = (\mathbf{c}.\mathbf{d}_1.cd.\mathsf{c}_1 \ldots \mathsf{c}_i)(\mathsf{root})_x$ for all $i \leq k$. Note that $a \notin \{c, d\}$, because $\mathbf{c}.\mathbf{d} \not\models F_a B$, so $(\mathbf{c}.\mathbf{d})(\mathsf{root})_a = (\mathbf{c}.\mathbf{d}')(\mathsf{root})_a$ holds as desired.

Now consider the guard ϕ_i associated with the call c_i where $i \leq k$. By assumption on P, ϕ_i is a propositional formula built out of the atomic formulas of the form $F_x S$ where $x \notin \{c, d\}$ is the agent making the call c_i. We already showed that $(\mathbf{c}.\mathbf{d}_1.\mathsf{c}_1 \ldots \mathsf{c}_{i-1})(\mathsf{root})_x = (\mathbf{c}.\mathbf{d}_1.cd.\mathsf{c}_1 \ldots \mathsf{c}_{i-1})(\mathsf{root})_x$, so the truth of these atomic formulas is not affected by the removal of the call cd from $\mathbf{d}$. This shows that we have $\mathbf{c}.\mathbf{d}_1.cd.\mathsf{c}_1 \ldots \mathsf{c}_{i-1} \models \phi_i$ iff $\mathbf{c}.\mathbf{d}_1.\mathsf{c}_1 \ldots \mathsf{c}_{i-1} \models \phi_i$ and so c_i can also be made by the protocol P after $\mathbf{c}.\mathbf{d}_1.\mathsf{c}_1 \ldots \mathsf{c}_{i-1}$. □

We establish now what the correctness of a propositional protocol implies for the order of calls.

Lemma 6 (Initiation). *Consider any call sequence* $\mathbf{c}$ *which is a prefix of a computation of a correct propositional gossip protocol* P *such that* $\mathbf{c} \models F_a B$*. There does not exist a call sequence* $\mathbf{d}$ *such that* $\mathbf{c}.\mathbf{d}.ab$ *is a prefix of* P*. (In other words, agent* a *will never call agent* b *if agent* a *already knows* B*.)*

Proof. Suppose such a call sequence $\mathbf{d}$ exists. If $\mathbf{c}.\mathbf{d}(\mathsf{root})_b \subseteq \mathbf{c}.\mathbf{d}(\mathsf{root})_a$, then we have $\mathbf{c}.\mathbf{d}(\mathsf{root})_a = \mathbf{c}.\mathbf{d}.ab(\mathsf{root})_a$ and so the guard ϕ of the call ab is still true after ab is made. As a result, ab could be repeated indefinitely after $\mathbf{c}.\mathbf{d}$; a contradiction with the assumption that P is terminating.

Otherwise, there exists a secret, X, such that $\mathbf{c}.\mathbf{d} \models F_b X \wedge \neg F_a X$ before the call ab takes place. Let us now remove all calls from $\mathbf{c}.\mathbf{d}$ that involve agents that are familiar with the secret X, which results in a call sequence $\mathbf{c}'.\mathbf{d}'$. Lemma 5 then implies that $\mathbf{c}'.\mathbf{d}'$ is also a prefix of P and $\mathbf{c}.\mathbf{d}(\mathsf{root})_a = \mathbf{c}'.\mathbf{d}'(\mathsf{root})_a$, so the call ab can still be made after $\mathbf{c}'.\mathbf{d}'$. At the same time, $\mathbf{c}'.\mathbf{d}'(\mathsf{root})_b \subsetneq \mathbf{c}.\mathbf{d}(\mathsf{root})_b$, because agent b is no longer familiar with secret X and possibly other secrets as well.

If there is still a secret Y left such that $\mathbf{c}'.\mathbf{d}' \models F_b Y \wedge \neg F_a Y$ then we again remove all calls from $\mathbf{c}'.\mathbf{d}'$ that involve agents that are familiar with the secret Y, which results in a call sequence $\mathbf{c}''.\mathbf{d}''$. We keep repeating this process until we reach a call sequence $\mathbf{c}^*.\mathbf{d}^*$ such that $\mathbf{c}^*.\mathbf{d}^*(\mathsf{root})_b \subseteq \mathbf{c}^*.\mathbf{d}^*(\mathsf{root})_a$ and $\mathbf{c}^*.\mathbf{d}^*.ab$ is a prefix of P, because $\mathbf{c}.\mathbf{d}(\mathsf{root})_a = \mathbf{c}^*.\mathbf{d}^*(\mathsf{root})_a$. Just like before, we arrive to a contradiction, because the call ab can now be repeated indefinitely. □

Already these two lemmas allow us to show non-existence of a correct propositional gossip protocol for a wide range of natural communication graph classes. The first graph class that we consider is the star graph, i.e., when communication is only possible via a single central agent. This was already shown in [9], however our proof is much more simplistic.

Theorem 7. *Suppose that the underlying undirected communication graph forms a star graph with at least 3 agents. No correct propositional protocol exists.*

Proof. Suppose such a protocol P exists. From Lemma 6 each agent, apart from the central agent, is involved in at most one call, as otherwise the protocol will not be correct. Therefore, the non-central agent involved in the first call will not have any further calls, and so will never become an expert. □

We now proceed to show that no correct propositional protocol exists when the underlying undirected communication graph is not complete. Note that if there are only two agents then this statement is trivial. In the case of three agents, a undirected connected graph with a missing link is a star graph so the statement follows from Theorem 7. The proof of this statement in the general case is quite complex, so we break it down into several lemmas. In all these lemmas, we make the assumption that a correct protocol P exists where there is no link between two agents denoted by a and b. Theorem 11 will later show how this assumption leads to a contradiction.

Lemma 8. *There exists a computation of P such that agent b learns A by receiving a call from another agent.*

Proof. We prove this by contradiction and so assume instead that in all computations agent b learns A by calling another agent.

Let us pick a computation where b knows the greatest number of secrets before learning A. In other words, if $\mathbf{c}.bc$ is a prefix of P where b learns A during the call bc, we require the size of $\mathbf{c}(\mathsf{root})_b$ to be the largest possible. This is well-defined as this value is an integer between 1 and $|\mathsf{A}|$ and agent b has learn to A in every computation as P is correct.

We know that $\mathbf{c} \models \neg F_b C$, because otherwise bc would not be possible due to Lemma 6. We can then remove all calls of agents familiar with C in $\mathbf{c}$ and obtain $\mathbf{c}'$. Due to Lemma 5, $\mathbf{c}'.bc$ is still a prefix of P. Note that c does not know A (nor any other secret apart from his own for that matter) after $\mathbf{c}'$, because all his calls were removed. At the same time we know that $\mathbf{c}(\mathsf{root})_b = \mathbf{c}'(\mathsf{root})_b$. If P is indeed correct, then it has to be possible to extend $\mathbf{c}'.bc$ to a prefix $\mathbf{c}'.bc.\mathbf{d}.bd$ of P, for some $\mathbf{d}$ and d, such that b finally learns A during bd. (Note that due to our original assumption, it cannot be db.) But then $\mathbf{c}(\mathsf{root})_b$ is smaller than $\mathbf{c}'.bc.\mathbf{d}(\mathsf{root})_b$, because the latter includes at least one more secret (namely C); this is a contradiction with the pick of the prefix $\mathbf{c}.bc$ of P as the one where b knows the most number of secrets before learning A. □

Lemma 9. *For any call sequence $\mathbf{c}$ without a-calls and any agent $c \in \mathsf{A} \setminus \{a, b\}$, if $\mathbf{c}.ca$ is a prefix of P such that $\mathbf{c} \models F_c B$ (i.e., a learns B from c), then $\mathbf{c}.\mathbf{d}.bc$ is also a prefix of P for some call sequence $\mathbf{d}$.*

Proof. Let us pick any prefix of P $\mathbf{c}.ca$ where $\mathbf{c}$ is without a-calls, such that $\mathbf{c} \models F_c B$. Note that after $\mathbf{c}.ca$, all agents that know A (agents a and c, only) also know B. As in every call all secrets are exchanged, any extension of $\mathbf{c}.ca$ would also have this property.

Due to Lemma 6, no agent that knows A would call b after $\mathbf{c}.ca$, because he already knows B. Moreover, b will never call a (missing link) and let us assume she does not call c either. Then, as b must learn A eventually, she must call a different agent. From here the proof follows similarly as in Lemma 8, but with an initial call sequence $\mathbf{c}.ca$.

Let us pick a computation that starts with $\mathbf{c}.ca$ where b knows the greatest number of secrets before learning A. In other words, if $\mathbf{c}.ca.\mathbf{d}.bd$ is a prefix of P where b learns A during the call bd for some $d \in \mathsf{A} \setminus \{a, b, c\}$, we require the size of $(\mathbf{c}.ca.\mathbf{d})(\mathsf{root})_b$ to be the largest possible. This is well-defined as this value is an integer between 1 and $|\mathsf{A}|$.

We know that $\mathbf{c}.ca.\mathbf{d} \models \neg F_b D$, because otherwise bd would not be possible due to Lemma 6. We can then remove all calls of agents familiar with D in $\mathbf{d}$ and obtain $\mathbf{d}'$. Due to Lemma 5, $\mathbf{c}.ca.\mathbf{d}(\mathsf{root})_b = \mathbf{c}.ca.\mathbf{d}'(\mathsf{root})_b$, so $\mathbf{c}.ca.\mathbf{d}'.bd$ is also a prefix of P.

Note that d does not know A after $\mathbf{c}.ca.\mathbf{d}'$, because $\mathbf{c}$ and $\mathbf{d}'$ have no calls involving agents familiar with A. Hence $\mathbf{c}.ca.\mathbf{d}'.bd \models \neg F_b A$. So if P is indeed correct, then it has to be possible to extend $\mathbf{c}.ca.\mathbf{d}'$ to a prefix $\mathbf{c}.ca.\mathbf{d}'.bd.\mathbf{e}.be$ of P, for some call sequence $\mathbf{e}$ and agent e, such that b finally learns A during be. (Note that it cannot be eb, because no agent familiar with A would call b after $\mathbf{c}.ca$.) But then $\mathbf{c}.ca.\mathbf{d}(\mathsf{root})_b$ is smaller than $\mathbf{c}.ca.\mathbf{d}'.bd.\mathbf{e}(\mathsf{root})_b$, because the latter includes at least one more secret (namely D); this is a contradiction with the pick of the prefix $\mathbf{c}.ca.\mathbf{d}.bd$ of P as the one where b knows

the most number of secrets before learning A. In conclusion, it must be possible for b to call c after $\mathbf{c}.ca$.

We have shown so far that $\mathbf{c}.ca.\mathbf{d}.bc$ has to be a prefix of P for some call sequence $\mathbf{d}$. Note that $\mathbf{c}.ca.\mathbf{d} \not\models F_bC$ due to Lemma 6. We can then remove all calls of agents familiar with C from the suffix $ca.\mathbf{d}$ of $\mathbf{c}.ca.\mathbf{d}$, to obtain $\mathbf{c}.\mathbf{d}'$. Then due to Lemma 5, $\mathbf{c}.\mathbf{d}'.bc$ is also a prefix of P. □

Using very similar techniques we can show the following.

Lemma 10. *For any call sequence* $\mathbf{c}$ *without* a*-calls and any agent* $c \in \mathsf{A} \setminus \{a, b\}$*, if* $\mathbf{c}.ca$ *is a prefix of* P *such that* $\mathbf{c} \models F_cB$*, and* d *is the last agent in a call with* c *before call* ca *takes place, then* $\mathbf{c}.ca.\mathbf{d}.da$ *is also a prefix of* P *for some* $\mathbf{d}$*.*

We now have all the ingredients needed to prove the main result of this section.

Theorem 11. *Suppose that the underlying undirected communication graph is not complete. No correct propositional protocol exists.*

Proof. Suppose such a correct propositional protocol P exists and there are two agents, say a and b, which cannot call each other.

From Lemma 8 there exists an agent $c \in \mathsf{A} \setminus \{a, b\}$ and a prefix $\mathbf{c}.ca$ of P such that $\mathbf{c} \models F_cB$. We know that $\mathbf{c} \not\models F_cA$, because otherwise ca would not be possible due to Lemma 6. We can then remove all calls of agents familiar with A in $\mathbf{c}$ and obtain $\mathbf{c}'$. Due to Lemma 5, $\mathbf{c}(\mathsf{root})_c = \mathbf{c}'(\mathsf{root})_c$, so $\mathbf{c}'.ca$ is also a prefix of P, with a not yet having been in a call until ca takes place.

Since $\mathbf{c}'.ca$ is a prefix of P where $\mathbf{c}'$ has no a-calls and $\mathbf{c}' \models F_cB$ then from Lemma 9 we get that $\mathbf{c}'.\mathbf{d}.bc$ is also a prefix of P for some $\mathbf{d}$. Therefore, $\mathbf{c}'$ cannot have bc nor cb call due to Lemma 6.

Note that there has to be at least one c-call in $\mathbf{c}'$, because $\mathbf{c}' \models F_cB$. We already excluded bc and cb. It cannot be ac nor ca either as ca takes place after $\mathbf{c}'$. Therefore, the last agent to be in a c-call in $\mathbf{c}'$ is some $d \in \mathsf{A} \setminus \{a, b, c\}$. From Lemma 10 we get that $\mathbf{c}'.ca.\mathbf{e}.da$ must also be a prefix of P for some $\mathbf{e}$. Note that also $\mathbf{c}' \models F_dB$, because when the call between d and c takes place, c already has to know B.

We know that $\mathbf{c}'.ca.\mathbf{e} \models \neg F_dA$, because otherwise da would not be possible due to Lemma 6. We can then remove all calls of agents familiar with A in $\mathbf{e}$ and obtain $\mathbf{e}'$. Due to Lemma 5, $\mathbf{c}'.ca.\mathbf{e}(\mathsf{root})_d = \mathbf{c}'.ca.\mathbf{e}'(\mathsf{root})_d$, so $\mathbf{c}'.ca.\mathbf{e}'.da$ is also a prefix of P. As $\mathbf{e}$ has had all calls to agents familiar with A removed, $\mathbf{e}'$ contains no c-calls. Hence, $\mathbf{c}'(\mathsf{root})_c = \mathbf{c}'.\mathbf{e}'(\mathsf{root})_c$, and so $\mathbf{c}'.\mathbf{e}'.ca$ is also a prefix of P. (As $\mathbf{e}'$ contains no call to or from agents familiar with A, $\mathbf{e}'$ can now occur before ca, because none of the calls can involve c or a.) As ca does not change the set of secrets known by d, and da does not change the set of secrets known by c, we get that both $\mathbf{c}'.\mathbf{e}'.ca.da$ and $\mathbf{c}'.\mathbf{e}'.da.ca$ are also prefixes of P.

As $\mathbf{c}'.\mathbf{e}'.ca$ is prefix of P, $\mathbf{c}'.\mathbf{e}'$ does not have any a-calls and $\mathbf{c}'.\mathbf{e}' \models F_cB$ then from Lemma 9 we get that $\mathbf{c}'.\mathbf{e}'.\mathbf{f}.bc$ is also a prefix of P for some $\mathbf{f}$.

We know that $\mathbf{c}'.\mathbf{e}'.\mathbf{f} \models \neg F_bC$, because otherwise bc would not be possible due to Lemma 6. We can then remove all calls of agents familiar with C in $\mathbf{f}$ and obtain $\mathbf{f}'$. Due to Lemma 5, $\mathbf{c}'.\mathbf{e}'.\mathbf{f}(\mathsf{root})_b = \mathbf{c}'.\mathbf{e}'.\mathbf{f}'(\mathsf{root})_b$, so $\mathbf{c}'.\mathbf{e}'.\mathbf{f}'.bc$ is also a prefix of P.

Note that $\mathbf{c}'.\mathbf{e}'.\mathbf{f}'.bc(\mathsf{root})_d = \mathbf{c}'.\mathbf{e}'(\mathsf{root})_d$, because all calls of agents familiar with C were removed from $\mathbf{f}'$ and d is familiar with C already after $\mathbf{c}'$. Hence, $\mathbf{c}'.\mathbf{e}'.\mathbf{f}'.bc.da$ is also a prefix of P, because da can take place immediately after $\mathbf{c}'.\mathbf{e}'$.

Now due to Lemma 9 we get that $\mathbf{c}'.\mathbf{e}'.\mathbf{f}'.bc.da.\mathbf{g}.bd$ is also a prefix of P, because $\mathbf{c}'.\mathbf{e}'.\mathbf{f}'.bc$ does not have any a-calls and $\mathbf{c}'.\mathbf{e}'.\mathbf{f}'.bc \models F_dB$, because $\mathbf{c}' \models F_dB$. We now get a contradiction with Lemma 6, because in this prefix b calls d even though b already knows D after the bc call in $\mathbf{c}'.\mathbf{e}'.\mathbf{f}'.bc.da.\mathbf{g}.bd$. □

4 Minimal number of calls

In this section we establish a lower bound on the number of calls needed for a propositional protocol to terminate in a state were all agents are experts. First, we start with one very useful observation.

Lemma 12 (Conversation). *For a protocol on n agents to correctly terminate in m calls, every agent must be involved in a call after at most $m - n + 2$ calls. Furthermore, after $m - n + p$ calls, each secret must be known by at least p agents.*

Proof. For an agent a and its secret A, each call can increase the number of agents that know A by at most 1. If a has not yet been involved in any calls, then the only agent which knows A is a. If after $m - n + 2$ calls, a has not yet been involved in a call, then a is the only agent which knows A. However, only $n - 2$ calls remain for $n - 1$ agents to learn A, which is impossible.

Similarly, if after $m - n + p$ calls, fewer than p know A, then $n - p$ calls remain for at least $n - p + 1$ agents to learn A. Again this is impossible as at most 1 agent can learn A in each call. □

We are now ready to partially resolve Problem 6 from [9] for the special case of propositional protocols.

Theorem 13. *No correct proposition protocol on n agents exists with fewer than $2n - 3$ calls.*

Proof. Let us assume a correct propositional protocol P exists which always terminates after at most $2n - 4$ calls.

First, Lemma 5 in [9] shows that every gossip protocol has a computation that starts with the same agent being involved in its first two calls. By relabelling the names of the agents, we can assume that we have a call between a and b, followed by a call between b and c. Wlog we can assume that these two calls are $ab.bc$, because the resulting gossip situation is always (AB, ABC, ABC) for a, b, c, respectively, and all other agents know just their own secret.

We claim that there must be a prefix of P of the form $ab.bc.\mathbf{c}.ac$ (or equivalently $ab.cb.\mathbf{c}.ac$) where $\mathbf{c}$ does not involve agents familiar with C. First of all, a has to learn C eventually. From Lemma 6, we know that after $ab.bc$ agent c will not call a, because he already knows A. Furthermore, due to Lemma 6, no agent that will learn C later will initiate a call with a, because he will learn A at the same time as C. So the only option left is that a learns C by calling another agent. Let $ab.bc.\mathbf{d}.ad$ be such a prefix of P

where a learns C from d. Clearly d cannot be b due to Lemma 6 and if d is c we are done. Thanks to Lemma 5 we can remove all calls in **d** that involve agents familiar with C and get a new prefix $ab.bc.\mathbf{d'}.ad$ of P after which a still does not know C. Therefore, there has to be an extension $ab.bc.\mathbf{d'}.ad.\mathbf{e}.ae$ of this prefix after which a learns C. If e is c we are done. Otherwise, we again remove all calls in **e** that involve agents familiar with C. We continue this process until finally a calls c. This has to happen eventually, because a can call each agent at most once and a has to learn C at some point.

Now, consider any prefix $ab.bc.\mathbf{c}.ac$ of P where **c** does not involve agents familiar with C, or any agents familiar with a secret which a is not familiar with after ac. This can be done by repeated use of Lemma 5 on **c** as necessary.

The length of **c** can be between 0 and $n-4$, by Lemma 12. If more calls were in **c** then after $n-1$ calls only two agents would know C, leaving at most $n-3$ calls for $n-2$ agents to learn C. This implies not every agent can be involved in **c**.

Let the number of calls in **c** be p. At most p agents (not including a) can be involved in **c**. Hence, after ac, at most $p+3$ agents have been involved in a call, after $p+3$ calls. This leaves $n-p-5$ calls for all remaining agents to have been involved in a call, with at least $n-p-3$ agents still to have a call.

From here, if we take every call either directly from or directly to this connected component which we shall now refer to a CC. These calls must be made and be available without any calls between two agents not in CC. Hence, at most 1 extra agent can be involved in a call for each of these call. Therefore, we can say that after $q+3$ calls, we have $n-q-5$ calls remaining before all must have been in a call with at least $n-q-3$ agents still to have a call. We shall now refer to all agents yet to be in a call when there are no calls left directly to or from CC as NCC. The next call then must be between two NCC agents (if any such agents exist).

If NCC is empty, then either the last agent called was called after n calls and hence we are done by Lemma 12, or ac was the final call, in which case after n calls only three agents know C after n calls, and we are also done by Lemma 12.

Assume then that NCC is non-empty. We know that the next call cannot be directly involved with CC, hence the call must be between two agents in NCC, say a' and b'. If a' or b' can now make a call to CC, then we have added 2 agents and their secrets to CC, whilst having 2 extra calls, so we are in the same situation and can repeat. If we ever end up with 2 or fewer agents in NCC then by Lemma 12 this can no longer be completed in $2n-4$ calls, as at least $n-2$ calls will have now taken place.

Again, due to Lemma 5 in [9] it must be possible for the next two calls between agents in NCC to involve the same agent (denoted by $\bar{b}$). If this was not the case then the protocol would either terminate, or these agents would be communicating with CC, as NCC is initially totally disconnected. Let $\bar{a}$ and $\bar{c}$ be the other agents involved. By relabelling the names of the agents, we can assume that these two calls are $\bar{a}\bar{b}.\bar{b}\bar{c}$. We now repeat the process as for CC, noting that $\bar{a}$ must call $\bar{c}$ eventually. This includes another r calls involving at most r agents and their secrets. By repeating this argument we get the desired result. □

We can even strengthen our lower bound further in the case of 4 agents.

Theorem 14. *No correct proposition protocol for 4 agents exists with fewer than 6 calls.*

5 Decision Problems for Propositional Protocols

We now move on to analysing the computation complexity of important decision problems for propositional gossip protocols such as checking termination, partial correctness and correctness. We say a protocol terminates if all computations are finite, i.e., there is no way for the scheduler to force the protocol to make an infinite number of calls. We first establish the necessary and sufficient condition for a propositional protocol to terminate.

Lemma 15. *A protocol will not terminate if and only if a call is made without the caller gaining new information within the first* $|\mathsf{A}|^2$ *calls.*

Proof. ($\Leftarrow$) Consider a propositional gossip protocol P. Let $\mathbf{c}.ab$ be a prefix of P such that $\mathbf{c}(\mathsf{root})_a = \mathbf{c}.ab(\mathsf{root})_a$. Hence there exists a rule $\psi \to ab$ for agent a such that $\mathbf{c} \models \psi$. But we clearly have then $\mathbf{c}.ab \models \psi$, because ψ only depends on the secrets agent a knows and they do not change after ab is made. Therefore, call ab can be performed again after $\mathbf{c}.ab$. It is easy to see that $\mathbf{c}(\mathsf{root})_a = \mathbf{c}.(ab)^k(\mathsf{root})_a$ and that $\mathbf{c}.(ab)^k$ is a prefix of P for any k, so P may not terminate.

($\Rightarrow$) Consider any infinite computation $\mathbf{c}$ of P. Along this computation, the current gossip situation can change at most $|\mathsf{A}|^2$ times, because each agent can be familiar with at most A secrets. So within the first $|\mathsf{A}|^2$ calls of $\mathbf{c}$ there has to be a call after which the caller does not learn any new secrets. □

As we will see, the previous lemma suffices to establish that all the decision problems studied in this section are in co-NP. To show them to be co-NP-hard, we show three different but similar reductions from the well-known 3-SAT problem.

Theorem 16. *The problem of checking if a given propositional gossip protocol* P *terminates is co-NP-complete.*

Proof (sketch). First, we show the problem to be in co-NP. Due to Lemma 15, to show *non-termination*, it suffices to guess a call sequence $\mathbf{c}$ of length $|\mathsf{A}|^2$ and a rule $\psi \to ab$ of P such that $\mathbf{c} \models \psi$ and $\mathbf{c}(\mathsf{root})_a = \mathbf{c}.ab(\mathsf{root})_a$. All of that is of polynomial size and can be checked in polynomial time.

To show the problem is co-NP-hard we will create a polynomial time reduction from the 3-SAT problem, such that termination's NO instances match with 3-SAT's YES instances. The basic idea of this is to have an agent which will become an expert iff the original problem is satisfiable. Once this agent becomes an expert, the scheduler can make that call indefinitely.

One final agent, f, is created. Three agents are created for each variable, one for true (true agent), one for false (false agent), and one to decide the truth assignment to this variable (trigger agent). One agent is created for each variable for each clause, to pass on information for the option above not chosen to the final agent (loser agents). One agent is created for each variable for each clause, to pass on information for the option above chosen to relevant clauses (winner agents). One agent is created for each variable for each clause, to pass on information for the option chosen to the final agent (pass agents). One agent is created for each clause.

The protocol is now set up in such a way that the scheduler has very few options, and each agent has very few calls. For each trigger agent, t, while this agent only knows its own secret T, t wants to call agent tv (variable true) or agent tf (variable false) and the scheduler will choose which. This is essentially the same as determining if the variable is true or false. Whichever is called first, we will take the opposite. Whichever is called first will know only its only secret, and T. At this stage the agent knows it has lost, as it does not know its negation's secret. It still calls the same agents as if it has won, however, these become losers agents when they realise they do not know the extra secret. These now call f and terminate.

t now makes a second call to the winner, and terminates. The winner now knows secrets TV and F. It calls the winner agents in turn. Firstly, these agents call its unique pass agent, which then passes the secrets on to f. This is to ensure that f can become an expert even if a variable is not used to satisfy any clauses. There is one winner agent for every clause. A call is made to the particular clause agent if it will satisfy the clause. This would be easy to do in the set up, as we know what is needed to satisfy each clause. Once a clause is satisfied, it will initiate a call with f. This call is only made once. If we assume the scheduler wants the protocol to not terminate, we just need 3 functions, one for each variable (and accompanying secrets), however, as there are only 7 permutations, we can include all of these to ensure the clause calls f in any circumstance.

Now, f will learn all secrets, apart from clause secrets in any scenario, but will only learn a clause secret if that clause is satisfied. Therefore, f becomes an expert iff all clauses are satisfied. f only makes a call if it becomes an expert, and trivially once this call is made the protocol will never terminate, as it can be repeated indefinitely. It is easy to see that this whole construction can be done in polynomial time and size. □

Using similar techniques we can show the other two problems are co-NP-complete.

Theorem 17. *The problem of checking if a given propositional gossip protocol P is partially correct is co-NP-complete.*

Theorem 18. *The problem of checking if a given propositional gossip protocol P is correct is co-NP-complete.*

Experimental evaluation. With the knowledge that the correctness check for a propositional gossip protocol is co-NP-complete, we ran experiments using NUXMV [13,11] in order to see for how many agents a computer can solve this problem in a reasonable amount of time. The experiments were run on an OMEN by HP Laptop PC - 15-ax000na (ENERGY STAR), with Intel® Core™ i5-6300HQ (2.3 GHz, up to 3.2 GHz, 6 MB cache, 4 cores) microprocessor and 8 GB DDR4-2133 SDRAM (2 x 4 GB) memory.

Experiments were carried out on LNS protocol, which would return a positive result, and LNS with a single link missing between two agents, which would return a negative result. We simulate the behaviour of the LNS gossip protocol with several optimization as NUXMV processes.

For the correct LNS protocol on 3 and 4 agents the results were almost instant, however on 5 agents results took 4 minutes, rising to over an hour and a half by 6 agents. At the same time, running the program on the incorrect LNS (when one edge was

removed) on 6 agents gave a result in 9 minutes. This suggests that on large protocols simply running a model checker on the direct encoding of the gossip protocol is not a practical algorithm for checking its correctness.

6 Conclusions

In this paper we solved several open problems about propositional gossip problems proposed in [9], but many interesting questions remain open. One is to further increase the lower bound on the minimal number of calls needed by a correct propositional protocol. No linear upper bound is known at the moment (the $2n - 3$ upper bound from [9] applies to general gossip protocols only). Another is to study simulation and bisimulation between such protocols as proposed in [9]. Finally, finding a practical correctness checking algorithm for propositional protocols would be a challenge as we established its co-NP-hardness.

Acknowledgments

We would like to thank the anonymous reviewers whose comments helped to improve this paper. The first author was supported by EPSRC NPIF PhD studentship. The second author was supported by EPSRC grant EP/P020909/1.

References

1. C. J. Anderson, N. Foster, A. Guha, J. Jeannin, D. Kozen, C. Schlesinger, and D. Walker. NetKAT: semantic foundations for networks. In *The 41st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '14*, pages 113–126. ACM, 2014.
2. K. R. Apt, D. Grossi, and W. van der Hoek. Epistemic protocols for distributed gossiping. In *Proceedings of the 15th Conference on Theoretical Aspects of Rationality and Knowledge (TARK 2015)*, volume 215 of *EPTCS*, pages 51–66, 2016.
3. K. R. Apt, D. Grossi, and W. van der Hoek. When are two gossips the same? In G. Barthe, G. Sutcliffe, and M. Veanes, editors, *LPAR-22. 22nd International Conference on Logic for Programming, Artificial Intelligence and Reasoning*, volume 57 of *EPiC Series in Computing*, pages 36–55. EasyChair, 2018.
4. K. R. Apt, E. Kopczyński, and D. Wojtczak. On the computational complexity of gossip protocols. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017*, pages 765–771, 2017.
5. K. R. Apt and D. Wojtczak. On decidability of a logic of gossips. In *Proceedings of the 15th European Conference, JELIA 2016*, volume 10021 of *Lecture Notes in Computer Science*, pages 18–33. Springer, 2016.
6. K. R. Apt and D. Wojtczak. Common knowledge in a logic of gossips. In *Proc. of the 16th Conference on Theoretical Aspects of Rationality and Knowledge (TARK 2017)*, volume 251 of *EPTCS*, pages 10–27, 2017.
7. K. R. Apt and D. Wojtczak. Decidability of fair termination of gossip protocols. In *Proc. of the 21st International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR 21)*, volume 1 of *Kalpa Publications in Computing*, pages 73–85, 2017.

8. K. R. Apt and D. Wojtczak. Verification of distributed epistemic gossip protocols. *J. Artif. Intell. Res. (JAIR)*, 62:101–132, 2018.
9. K. R. Apt and D. Wojtczak. Open problems in a logic of gossips. In *Proceedings Seventeenth Conference on Theoretical Aspects of Rationality and Knowledge (TARK 2019)*, volume 297 of *EPTCS*, pages 1–18, 2019.
10. 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.
11. A. Biere and R. Bloem, editors. *Computer Aided Verification - 26th International Conference, CAV 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 18-22, 2014. Proceedings*, volume 8559 of *Lecture Notes in Computer Science*. Springer, 2014.
12. R. T. Bumby. A problem with telephones. *SIAM Journal on Algebraic Discrete Methods*, 2(1):13–18, 1981.
13. R. Cavada, A. Cimatti, M. Dorigatti, A. Griggio, A. Mariotti, A. Micheli, S. Mover, M. Roveri, and S. Tonetta. The nuxmv symbolic model checker. In Biere and Bloem [11], pages 334–342.
14. M. C. Cooper, A. Herzig, F. Maffre, F. Maris, and P. Régnier. Simple Epistemic Planning: Generalised Gossiping. In *Proceedings of ECAI 2016*, volume 285 of *Frontiers in Artificial Intelligence and Applications*, pages 1563–1564. IOS Press, 2016.
15. M. C. Cooper, A. Herzig, F. Maris, and J. Vianey. Temporal epistemic gossip problems. In *European Conference on Multi-Agent Systems*, pages 1–14. Springer, 2018.
16. R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. Knowledge-based programs. *Distributed Computing*, 10(4):199–225, 1997.
17. M. Gattinger and J. Wagemaker. Towards an analysis of dynamic gossip in NetKAT. In *International Conference on Relational and Algebraic Methods in Computer Science*, pages 280–297. Springer, 2018.
18. F. Harary and A. J. Schwenk. The communication problem on graphs and digraphs. 1974.
19. 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.
20. A. Herzig and F. Maffre. How to share knowledge by gossiping. In *Proc of the 13th European Conference on Multi-Agent Systems (EUMAS 2015), Revised Selected Papers*, volume 9571, pages 249–263. Springer, 2015.
21. A. Herzig and F. Maffre. How to share knowledge by gossiping. *AI Communications*, 30(1):1–17, 2017.
22. J. Hromkovič, R. Klasing, A. Pelc, P. Ruzicka, and W. Unger. *Dissemination of Information in Communication Networks - Broadcasting, Gossiping, Leader Election, and Fault-Tolerance*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2005.
23. D. Kempe, A. Dobra, and J. Gehrke. Gossip-based computation of aggregate information. In *Proc. of the 44th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '03, pages 482–491. IEEE, 2003.
24. A. Kermarrec and M. van Steen. Gossiping in distributed systems. *Operating Systems Review*, 41(5):2–7, 2007.
25. R. Ladin, B. Liskov, L. Shrira, and S. Ghemawat. Providing high availability using lazy replication. *ACM Transactions on Computer Systems (TOCS)*, 10(4):360–391, 1992.
26. R. Tijdeman. On a telephone problem. *Nieuw Archief voor Wiskunde*, 3(XIX):188–192, 1971.
27. H. van Ditmarsch, D. Grossi, A. Herzig, W. van der Hoek, and L. B. Kuijer. Parameters for epistemic gossip problems. In *Proceedings of the 12th Conference on Logic and the Foundations of Game and Decision Theory (LOFT 2016)*, 2016.
28. H. van Ditmarsch and I. Kokkinis. The expected duration of sequential gossiping. In *Multi-Agent Systems and Agreement Technologies*, pages 131–146. Springer, 2017.

29. H. van Ditmarsch, I. Kokkinis, and A. Stockmarr. Reachability and expectation in gossiping. In B. An, A. Bazzan, J. Leite, S. Villata, and L. van der Torre, editors, *PRIMA 2017: Principles and Practice of Multi-Agent Systems*, pages 93–109, Cham, 2017. Springer International Publishing.
30. H. van Ditmarsch, W. van Der Hoek, and L. B. Kuijer. The logic of gossiping. *Artificial Intelligence*, 286:103306, 2020.
31. H. van Ditmarsch, J. van Eijck, P. Pardo, R. Ramezanian, and F. Schwarzentruber. Epistemic protocols for dynamic gossip. *J. of Applied Logic*, 20(C):1–31, 2017.
32. H. van Ditmarsch, J. van Eijck, P. Pardo, R. Ramezanian, and F. Schwarzentruber. Dynamic gossip. *Bull. Iran. Math. Soc.*, pages 1–28, 2018.