GoMoChe: Gossip Model Checking

(Extended Abstract for LAMAS & SR 2022, Rennes)

Malvin Gattinger
University of Amsterdam
malvin@w4eg.eu

ABSTRACT

The gossip problem provides a simple model of information exchange in distributed systems. For the analysis and verification of gossip protocols, logics have been developed, but manual computation and case analysis can be tedious and error-prone.

Here we present GoMoChe, a special-purpose model checker for gossip. The tool can decide when protocols known from the literature such as "Learn New Secrets" are successful. GoMoChe uses a version of epistemic logic with protocol-dependent knowledge and was developed to find and verify results in previous work.

The main version models a synchronous setting where agents always know how many calls happened, but we also provide an experimental version for asynchronous settings where agents only observe their own calls.

KEYWORDS

Gossip Protocols, Model Checking, Epistemic Logic

1 INTRODUCTION

The gossip problem, also known as the telephone problem, in its simplest form can be stated as follows:

> Suppose n agents initially each know a unique secret. The agents make phone calls in which they tell each other all secrets they know. How many calls are needed until everyone knows all secrets?

The classic result is that $2n - 4$ calls are necessary and sufficient, as for example shown in [6]. Many similar results, for example for non-total "phone book" graphs, can be found in [3].

Besides its combinatoric and graph theoretic appeal, the gossip problem also serves as a toy example of a distributed system and a model of information synchronisation between multiple agents, be they actual gossipers or nodes of a distributed database.

The classic $2n-4$ result assumes a central scheduler who decides which calls should be made in which order. In contrast, recent research is about a decentralized setting where gossipers decide on their own whom to call. This motivates the study of epistemic protocols, i.e. calling conditions which agents (can) use to make this decision. Common examples from the literature are *Learn New Secrets*, LNS ("Agent a may call b iff a does not know the secret of b") or *Possible Information Growth*, PIG ("Agent a may call b iff a considers it possible that a or b will learn a new secret in that call"). For a precise analysis of these protocols, logics have been developed [1, 8, 10, 12]. Some also consider *dynamic* gossip from [11], where agents exchange phone numbers in addition to secrets, making the graph grow while calls are made.

Epistemic logic is a powerful tool to analyse distributed (dynamic) gossip protocols. In particular it allows us to study the (higher-order) knowledge obtained by gossiping agents, and answer questions such as the following.

(1) After the call sequence $ab; bc; ac$, does agent a know that agent b knows the secret of agent c?
(2) Is the call sequence $ab; cd; ac; bd$ successful, i.e. do all agents know all secrets afterwards? Moreover, is it super successful, i.e. do all agents know that all agents know all secrets?
(3) Given the gossip graph in the left part of Figure 1, how many LNS sequences are (un)successful?
(4) After the sequence $ab; bc; cd; bd$, does agent a know that if they call agent b then b will tell a the secret d?

All these questions can be made precise and then be answered using epistemic logic. However, doing the necessary case analysis and computation manually with pen and paper can be quite tedious.

Here we present *GoMoChe*, a special-purpose model checker for the analysis of gossip protocols. The tool was developed for and has been used to find and verify part of the results in [5, 7, 8].

GoMoChe is available at https://github.com/m4lvin/GoMoChe and is free software under the GNU General Public License 3. [1]

2 DYNAMIC GOSSIP: BASIC DEFINITIONS

We only provide basic definitions here and refer to [8] for more details and precision.

Definition 2.1. A *gossip graph* is a tuple $G = (A, N, S)$ where A is a finite set of agents, $N \subseteq A \times A$ is called the *number* relation, $S \subseteq A \times A$ is called the *secret* relation. A gossip graph is *initial* when S is the identity.

Definition 2.2. A *call* is a pair of agents $(x, y) \in A \times A$ which we also denote by xy. We denote finite *sequences* of calls by σ and the empty sequence by ϵ. The result of executing a call on G where $(x, y) \in N$ is $G^{xy} := (A, N^{xy}, S^{xy})$ where $N^{xy} := N \cup \{(x, z) \mid (y, z) \in N\} \cup \{(y, z) \mid (x, z) \in N\}$ and S^{xy} is defined analogously. We denote finite sequences of calls by σ and the empty sequence by ϵ. Executing a sequence σ on G is defined inductively to obtain $G^{\sigma} = (A, N^{\sigma}, S^{\sigma})$.

Example 2.3. We show two gossip graphs in Figure 1.

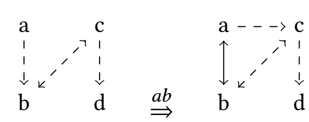

Figure 1: An initial gossip graph, a call and the resulting graph. Dashed lines show N, solid lines show $N \cap S$.

[1] This PDF file was last updated on 2022-07-22. The latest version is available at https://malv.in/2022/LAMASSR-GoMoChe.pdf.

3 PROTOCOL-DEPENDENT KNOWLEDGE

A key part of GoMoChe is a recursive model checking algorithm for the following language and semantics. The definitions below are mutually inductive and the reader might worry whether this is well-founded. We refer to [8] for explanations and proofs, and only note here that we do not allow self-referential protocols. That is, a protocol P may not use the operator K_i^P.

Definition 3.1. Let i range over A and let P be a protocol from Definition 3.2. The *language of protocol-dependent knowledge* is:

$$\varphi ::= \top \mid N_i i \mid S_i i \mid C_i i \mid i = i \mid \neg\varphi \mid \varphi \wedge \varphi \mid K_i^P \varphi \mid [\pi]\varphi$$

$$\pi ::= ?\varphi \mid ii \mid \pi;\pi \mid \pi \cup \pi \mid \pi^*$$

Besides this, $\bot$, $\vee$ and $\rightarrow$ are defined as usual and we define $\langle\pi\rangle\varphi := \neg[\pi]\neg\varphi$. We also define $\hat{K}_i^P\varphi := \neg K_i^P\varphi$ for "Agent i, given P, considers it possible that φ". In the implementation all connectives are primitives, to reduce the number of recursive calls. Moreover, GoMoChe provides quantifiers to say "For all agents x" etc.

Definition 3.2. A *protocol* is a function P assigning to each pair of agents a formula called the protocol condition P_{ab}.

Example 3.3. The *Learn New Secrets* (LNS) protocol is given by $LNS_{ab} := \neg S_a b$. The soft look-ahead strengthening of LNS is given by $LNS^{\blacklozenge} := LNS_{ab} \wedge \hat{K}_a^{LNS}[ab]\langle P\rangle \bigwedge_{x,y} S_x y$. (See [8] for more.)

Definition 3.4. A *state* is a tuple (G, σ) where G is an initial gossip graph and σ is a sequence executable on G. The semantics on gossip states are given by standard Boolean semantics and

$$\begin{array}{lll} G, \sigma \vDash N_x y & :\Leftrightarrow & (x, y) \in N^\sigma \\ G, \sigma \vDash S_x y & :\Leftrightarrow & (x, y) \in S^\sigma \\ G, \sigma \vDash C_x y & :\Leftrightarrow & xy \in \sigma \text{ or } yx \in \sigma \\ G, \sigma \vDash x = y & :\Leftrightarrow & x = y \\ G, \sigma \vDash K_a^P \varphi & \text{iff} & G, \sigma' \vDash \varphi \text{ for all } (G, \sigma') \sim_a^P (G, \sigma) \\ G, \sigma \vDash [\pi]\varphi & \text{iff} & G, \sigma' \vDash \varphi \text{ for all } (G, \sigma') \in [\![\pi]\!](G, \sigma) \end{array}$$

with $[\![\pi]\!](G, \sigma) = \{(G, \sigma') \mid ((G, \sigma), (G, \sigma')) \in [\![\pi]\!]\}$ defined by

$$\begin{array}{rcl} [\![?\varphi]\!](G, \sigma) & := & \{(G, \sigma) \mid G, \sigma \models \varphi\} \\ [\![ab]\!](G, \sigma) & := & \{(G, (\sigma; ab)) \mid G, \sigma \models N_a b\} \\ [\![\pi; \pi']\!](G, \sigma) & := & \bigcup\{[\![\pi']\!](G, \sigma') \mid (G, \sigma') \in [\![\pi]\!](G, \sigma)\} \\ [\![\pi \cup \pi']\!](G, \sigma) & := & [\![\pi]\!](G, \sigma) \cup [\![\pi']\!](G, \sigma) \\ [\![\pi^*]\!](G, \sigma) & := & \bigcup\{[\![\pi^n]\!](G, \sigma) \mid n \in \mathbb{N}\}. \end{array}$$

If $G, \sigma \models P_{ab}$ we say that ab is *P-permitted* at (G, σ). A *P-permitted call sequence* consists of P-permitted calls.

Definition 3.5. We define $\sim_a^P$ for agent a and protocol P between states (G, σ) by induction on σ where ϵ is the empty sequence.

- $(G, \epsilon) \sim_a^P (G, \epsilon)$;
- if $(G, \sigma) \sim_a^P (G, \tau)$, $N_b^\sigma = N_b^\tau$, $S_b^\sigma = S_b^\tau$, and $G, \sigma \models P_{ab}$ and $G, \tau \models P_{ab}$, then $(G, \sigma; ab) \sim_a^P (G, \tau; ab)$;
  if $(G, \sigma) \sim_a^P (G, \tau)$, $N_b^\sigma = N_b^\tau$, $S_b^\sigma = S_b^\tau$, and $G, \sigma \models P_{ba}$ and at $G, \tau \models P_{ab}$, then $(G, \sigma; ba) \sim_a^P (G, \tau; ba)$;
- if $(G, \sigma) \sim_a^P (G, \tau)$ and $a \notin \{c, d, e, f\}$ such that $G, \sigma \models P_{cd}$ and $G, \tau \models P_{ef}$, then $(G, \sigma; cd) \sim_a^P (G, \tau; ef)$.

In GoMoChe Defintion 3.5 is given by the function `epistAlt` of type `Agent -> Protocol -> State -> [State]` in the `Gossip.General` module.

4 USAGE AND FEATURES

GoMoChe is implemented in Haskell and used via `ghci`, the interactive Haskell compiler. To answer the questions from the introduction we can use GoMoChe as follows.

```
-- (1)
> eval (totalInit 3, parseSequence "ab;bc;ac") (K 0 anyCall (S 1 2))
True
-- (2)
> isSuccSequence (totalInit 4,[]) (parseSequence "ab;cd;ac;bd")
True
> isSuperSuccSequence lns (totalInit 4,[]) (parseSequence "ab;cd;ac;bd")
False
-- (3)
> statistics lns (parseGraph "01-12-231-3 I4",[])
(57,20)
-- (4)
> eval (totalInit 4, parseSequence "ab;bc;cd;bd")
       (K 0 anyCall (Box (Call 0 1) (S 0 3)))
False
> eval (totalInit 4, parseSequence "ab;bc;cd;bd")
       (K 0 lns (Box (Call 0 1) (S 0 3)))
True
```

The last two queries show that the answer depends on which protocol agent a (also denoted by 0) assumes: Agent a knows it if they assume `lns` is used, but do not know it when `anyCall` is allowed.

Other notable functions in GoMoChe include `dispDot` to visualise gossip graphs and execution trees, and `knowledgeOverview` to generate overview tables as shown in [9].

5 CONCLUDING REMARKS

Related Work. Parts of GoMoChe are inspired by the epistemic model checkers DEMO and SMCDEL [2, 13]. A similar tool developed recently by Ramon Meffert is *ElmGossip* from [4]. In contrast to GoMoChe, the ElmGossip tool offers a graphical user interface and is meant as an easy to use tool for students and researchers. While GoMoChe is meant to be used in ghci or as a Haskell library, ElmGossip does not require the user to be familiar with Haskell or any other programming language. On the other hand, ElmGossip does not keep track of higher-order knowledge and is not a model checker.

Asynchronous semantics. The main version of GoMoChe implements synchronous semantics as in Definition 3.5. In an experimental version (in the `async` branch) we implemented asynchronous semantics where for example $ab; bc; cd; ab \sim_a ab; dc; ab$. However, agents then often consider infinitely many sequences possible, hence we currently enforce a maximum length of call sequences. This ad-hoc solution is not sound, but suffices to find and prove negative results, i.e. to show that agents do *not* know something. In the future we hope to use reduction techniques from [10] to obtain finitary representations.

Future work. We plan to further improve the usability and documentation of GoMoChe. Other variants of gossip to consider include broadcast calls and unreliable agents.

Acknowledgments. I thank my coauthors in [5, 7, 8] for useful suggestions and catching bugs during the development of GoMoChe.

REFERENCES

[1] K.R. Apt and D. Wojtczak. 2018. Verification of Distributed Epistemic Gossip Protocols. *J. Artif. Intell. Res.* 62 (2018), 101–132. https://doi.org/10.1613/jair.1.11204

[2] Malvin Gattinger. 2022. SMCDEL — An Implementation of Symbolic Model Checking for Dynamic Epistemic Logic with Binary Decision Diagrams. https://github.com/jrclogic/SMCDEL

[3] Sandra M Hedetniemi, Stephen T Hedetniemi, and Arthur L Liestman. 1988. A survey of gossiping and broadcasting in communication networks. *Networks* 18, 4 (1988), 319–349. https://doi.org/10.1002/net.3230180406

[4] Ramon Meffert. 2021. Tools for Gossip. https://fse.studenttheses.ub.rug.nl/23961/ BSc thesis, University of Groningen. Code available at https://github.com/RamonMeffert/elm-gossip.

[5] Rahim Ramezanian, Rasoul Ramezanian, Hans van Ditmarsch, and Malvin Gattinger. 2021. Everyone Knows that Everyone Knows. In *Mathematics, Logic, and Their Philosophies: Essays in Honour of Mohammad Ardeshir*, Mojtaba Mojtahedi, Shahid Rahman, and Mohammad Saleh Zarepour (Eds.). https://doi.org/10.1007/978-3-030-53654-1_5

[6] Robert Tijdeman. 1971. On a telephone problem. *Nieuw Archief voor Wiskunde* 3, 19 (1971), 188–192.

[7] Hans van Ditmarsch, Malvin Gattinger, Ioannis Kokkinis, and Louwe B. Kuijer. 2019. Reachability of Five Gossip Protocols. In *Proceedings of RP 2019*, Emmanuel Filiot, Raphaël Jungers, and Igor Potapov (Eds.). https://doi.org/10.1007/978-3-030-30806-3_17

[8] Hans van Ditmarsch, Malvin Gattinger, Louwe B. Kuijer, and Pere Pardo. 2019. Strengthening Gossip Protocols using Protocol-Dependent Knowledge. *Journal of Applied Logics - IfCoLog Journal of Logics and their Applications* 6, 1 (2019), 157–203. https://doi.org/10.48550/arXiv.1907.12321

[9] Hans van Ditmarsch, Malvin Gattinger, and Rahim Ramezanian. 2020. Everyone Knows that Everyone Knows: Gossip Protocols for Super Experts. *CoRR* abs/2011.13203 (2020). arXiv:2011.13203 https://arxiv.org/abs/2011.13203

[10] H. van Ditmarsch, W. van der Hoek, and L.B. Kuijer. 2020. The logic of gossiping. *Artificial Intelligence* 286 (2020), 103306. https://doi.org/10.1016/j.artint.2020.103306

[11] Hans van Ditmarsch, Jan van Eijck, Pere Pardo, Rahim Ramezanian, and François Schwarzentruber. 2017. Epistemic protocols for dynamic gossip. *Journal of Applied Logic* 20 (2017), 1–31. https://doi.org/10.1016/j.jal.2016.12.001

[12] Hans van Ditmarsch, Jan van Eijck, Pere Pardo, Rahim Ramezanian, and François Schwarzentruber. 2019. Dynamic Gossip. *Bulletin of the Iranian Mathematical Society* 45, 3 (2019), 701–728. https://doi.org/10.1007/s41980-018-0160-4

[13] Jan van Eijck. 2007. DEMO—a demo of epistemic modelling. In *Interactive Logic. Selected Papers from the 7th Augustus de Morgan Workshop, London*, Vol. 1. 303–362. https://homepages.cwi.nl/~jve/papers/07/pdfs/DEMO_IL.pdf