university of groningen / faculty of science and engineering / bernoulli institute

# Tools for Gossip

Bachelor's Project Thesis

R.A. Meffert, s2702207, r.a.meffert@student.rug.nl
Supervisors: Dr. B.R.M. Gattinger

**Abstract:**

## 1 Introduction

*Gossip protocols* are protocols that describe the way rumors—or, more generally, secrets—are shared in multi-agent environments. The goal of the protocols is to communicate all secrets to all agents. A lot of research has been done in this field, starting with research on the spread of infectious diseases (Kermack & McKendrick, 1927).

The definition of the gossip problem generally used nowadays was first introduced in 1972 by Hajnal et al.* In short, agents are represented as nodes in a graph, with the edges representing a "call"—that is, one agent transferring all of their secrets to another agent. When all agents can contact all other agents, Hajnal et al. proved that this can be done in $2n - 4$ calls, where $n$ is the number of agents.

The problem as formulated above requires the oversight of a central authority in order to know whether all agents know all secrets. However, there are many applications where this is not feasible or desirable†. Another problem is that it often cannot be guaranteed that all agents can contact all other agents. This has led to the sub-fields of *distributed gossip*, addressing the first issue, and *dynamic gossip*, addressing the second. The combination of these fields, where there is no overseer and not all agents can contact all other agents, is called *distributed dynamic gossip*.

This paper will describe a tool that has been developed to aid research in distributed dynamic gossip. It is able to visualise the connections between agents, allows exploring the execution tree of different (semi-arbitrary) protocols and validate call sequences given a graph and/or a protocol

### 1.1 Notation

This paper takes its notation from Van Ditmarsch et al. (2018) and Van Ditmarsch et al. (2019). The used notation is explained below.

*TODO: Tijdeman (1971) might have been earlier, but I have not been able to find a pdf of that paper. It is referenced in Van Ditmarsch et al. (2018) though.

†TODO: maybe find a citation for this? Or just explain it better

**Definition 1** (Gossip graph)**.** Let $A$ be a set of agents $\{a, b, \dots\}$. Two binary relations on $A$ are defined: $N, S \subseteq A^2$. The first denotes the *number* relation, the second the *secret* relation. A gossip graph $G$ is then defined as the a triple $(A, N, S)$.

**Definition 1.1** (Binary relation)**.** $Pxy$ says that agent $x$ has relation $P$ to agent $y$.
Formally: $(x, y) \in P$

**Definition 1.2** (Identity relation)**.** $I_A$ is the set of relations where all agents have a relation to themselves.
Formally: $\{(x, x) \mid x \in A\}$

**Definition 1.3** (Converse relation)**.** $P^{-1}$ is the set of relations in $P$, but with their direction switched.
Formally: $P^{-1} = \{(x, y) \mid Pyx\}$

**Definition 1.4** (Composition relation)**.** The composition of the relations $P$ and $Q$ is a new relation such that the tuple $(x, z)$ is in said new relation iff there exists another agent y such that $(x, y) \in P$ and $(y, z) \in Q$.
Formally: $P \circ Q = \{(x, z) \mid \exists y((x, y) \in P \wedge (y, z) \in Q)\}$

**Definition 1.5.** $P_x$ represents the agents $x$ has relation $P$ with.
Formally: $P_x = \{y \in A \mid Pxy\}$

**Definition 1.6.** $P^i$ represents the $i$th composition of relation $P$ with itself.
Formally:

$$P^i = \begin{cases} P & \text{for } i = 1 \\ P^{i-1} \circ P & \text{for } i > 1 \end{cases}$$

**Definition 1.7.** $P^*$ represents the set of binary relations that are obtained through repeated relation composition of $P$ with itself.
Formally:

$$\bigcup_{i \geq 1} P^i$$

### 1.2 Implementation

This project uses Elm, a statically typed functional programming language for web development with its roots in Functional Reactive Programming (Czaplicki & Chong, 2013). This has several advantages:

1. Implementing mathematical functions is more natural in functional languages because functions in Elm are pure;
2. Elm is compiled to vanilla Javascript, ensuring the application will work in virtually all browsers;
3. Elm does static type checking while compiling, ensuring type safety and thus reducing runtime errors;

Another advantage of using a functional language is that much of the notation introduced in section 1.1 can be translated fairly directly. For example, to evaluate protocol conditions, the last call in a call sequence needs to be checked. Where in mathematical notation this is represented as $\sigma_x = \tau ; xy$ ("the last call in the sequence of calls containing $x$ was a call made by $x$ to another agent"), this can be represented in Elm as follows:

```
lastFrom =
    case reverse sigma_x of
        [] ->
            False

        (from, to) :: tau ->
            from == x
```

This states that if $\sigma_x = \epsilon$, the condition is false. Otherwise, it checks to see if the last call was made by x, and returns true if that is the case.

# 2 Method

# References


Czaplicki, E., & Chong, S. (2013). Asynchronous functional reactive programming for GUIs. *SIGPLAN Not.*, *48*(6), 411–422. https://doi.org/10/f45mkb

Hajnal, A., Milner, E. C., & Szemerédi, E. (1972). A cure for the telephone disease. *Canadian Mathematical Bulletin*, *15*(3), 447–450. https://doi.org/10/cpr4cv

Kermack, W. O., & McKendrick, A. G. (1927). A contribution to the mathematical theory of epidemics. *Proceedings of the Royal Society of London*, *115*(772), 700–721. https://doi.org/10.2307/94815

van Ditmarsch, H., Gattinger, M., Kuijer, L. B., & Pardo, P. (2019). Strengthening gossip protocols using protocol-dependent knowledge. *Journal of Applied Logics*, *6*(1), 157–203.

van Ditmarsch, H., van Eijck, J., Pardo, P., Ramezanian, R., & Schwarzentruber, F. (2018). Dynamic gossip. *Bulletin of the Iranian Mathematical Society*, *45*(3), 701–728. https://doi.org/10/cvpm

# A Appendix