
# Epistemic Gossip Protocols

A thesis submitted in accordance with the
requirements of the University of Liverpool
for the degree of Doctor of Philosophy by

**Maduka Attamah**

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF LIVERPOOL

December 2015

To my father Hilary Oddo Attamah (1940 - 2013)

# Contents

# Illustrations

## List of Figures



## List of Tables

# Abstract

In this thesis we study epistemic protocols for gossip. Each agent in the gossip scenario knows a unique piece of information which is called a secret. Agents communicate with each other by means of pairwise telephone calls, and in each call the calling pair of agents exchange all the secrets they currently know. In an epistemic gossip protocol, an agent $a$ can call another agent $b$, not because it is so instructed, but because agent $a$ knows that it satisfies some knowledge-based condition defined by the protocol.

The goal of gossiping is typically epistemic, for example, that after a sequence of calls, every agent knows the secret of every other agent. The question then arises as to which knowledge conditions bring about the goal of gossiping, and what properties the resulting protocols have. In this thesis we describe a theoretical framework for the study of epistemic gossip protocols based on dynamic epistemic logic. We describe a number of epistemic gossip protocols and formalise these protocols using our theoretical framework. We study and prove the dynamic properties of these protocols in various types of underlying network topologies such as the line topology network, circle topology network, tree topology network, and the complete topology network.

Based on our theoretical framework, we implement a software framework for describing, modelling and checking the dynamic properties of epistemic gossip protocols. We call this software framework the Epistemic Gossip Protocol (EGP) tool. The EGP tool automates the checking of dynamic properties of a given epistemic gossip protocol, such as, whether the given protocol achieves the goal of gossiping for every execution sequence of the protocol, whether the given protocol can produce execution sequences that lead to a deadlock, or whether the given protocol can produce an infinite execution sequence due to a loop. We describe the details of the implementation of the EGP tool, and use the tool to model, and check the dynamic properties of our example protocols. We present and discuss the results obtained from our experiments with the EGP tool.

# Acknowledgements

I would like to express my immense gratitude to my supervisors, who have also been my teachers: Prof. Wiebe van der Hoek, Dr. Davide Grossi and Prof. Hans van Ditmarsch. I thank them for their constant support and guidance in the course of my studies.

There are other people who helped me greatly along this path, I thank them: Prof. Leszek Gąsieniec, Ken Chan, Bev Sinclair and Lyn Hughes. Big thanks to all the academic and administrative staff of the Department of Computer Science, University of Liverpool. Special thanks to Elaine Smith, Helen Mattocks, and Hannah Fosh.

I thank my examiners for accepting to take on the role. Special gratitude to Prof. John-Jules Meyer who had to travel all the way from Netherlands to grace the occasion, and special gratitude to Dr. Alexei Lisitsa who was once also a teacher of mine.

I acknowledge and thank the Commonwealth Scholarship Commission and the University of Liverpool, both of whom jointly funded my masters programme in Advanced Computer Science at the University of Liverpool. That masters programme prepared me for the Ph.D. research that followed. My special gratitude goes to the Department of Computer Science for awarding me the departmental studentship which funded my Ph.D. research.

I remember in a special way my father who departed this world in the course of my studies. A few days before his departure, his last words to me were words of encouragement for my Ph.D. research. I thank him for all his support, and express my joy in completing this work. I give special gratitude and affection also to my lovely mother who looked forward to and prayed for my success in this endeavour.

My friends and colleagues: Tobenna Ejiogu, Emmanuel Ifeh, Onema Adojoh, Latifa Al-Abdulkarim, Eric Schneider, Fr. Peter Haverty, Fr. Peter Cookson, Xavier Bosch, Richard Lea, Adam Walker, Christopher Griffiths, and Rory White. I have named only a few of those whose friendship added warmth to my years of study in the United Kingdom. To all of them I say a big thank you.

Finally, I give thanks to God for being my ultimate guide, provider and support.

*Maduka Attamah*
*Liverpool, December 2015*

# Chapter 1

# Introduction

The rise of ubiquitous computing through, for instance, handheld and wearable devices, smart appliances and sensor networks, comes with a growing demand for such devices to be able to communicate messages containing basic information. Think also of self-driving cars and robotic vehicles that scavenge new planets, disaster areas and other environments where direct human observation is difficult to achieve. These vehicles will need to communicate information like their position, velocity, weather and traffic conditions, sometimes to a central monitoring system or to an information hub, and sometimes directly to each other. In this thesis, we study protocols for such communications but within a very simple setting, namely, the gossip problem.

A well-studied phenomenon in network theory is that of optimal schedules to distribute information by one-to-one communication between nodes. One can take these communicative interactions to be 'telephone calls', and this process of spreading information is known as *gossiping* [30]. It is typical to assume a global scheduler who simply executes a possibly non-deterministic protocol. Such a protocol can be seen as a general scheme or a rule which generates a set of execution sequences of communicative interactions among the agents in the scenario, where such generated execution sequences adhere to the rule given by the protocol. When the rule given by a protocol is some knowledge condition regarding some agent or agents in the scenario, then such a protocol is *epistemic*. We investigate *distributed epistemic gossip protocols*[*], where an agent $a$ will call another agent not because it is so instructed but based on its knowledge or ignorance of the information that is distributed over the network.

This chapter describes the gossip problem and the motivation for this research work. We conclude the chapter with an overview of the thesis.

## 1.1 The Gossip Problem

Communication protocols have the aim to share knowledge between nodes in a predescribed way. Consider the following scenario.

[*]Throughout this thesis, we will use the terms 'distributed epistemic gossip protocol' and 'epistemic gossip protocol' interchangeably, since we always assume that each agent executes its own protocol.

TABLE 1.1: **Information transitions for the sequence:** $ab; cd; ac; bd$**.**

| | $a$ | $b$ | $c$ | $d$ |
|---|---|---|---|---|
| | $A$ | $B$ | $C$ | $D$ |
| $ab$ | $AB$ | $AB$ | $C$ | $D$ |
| $cd$ | $AB$ | $AB$ | $CD$ | $CD$ |
| $ac$ | $ABCD$ | $AB$ | $ABCD$ | $CD$ |
| $bd$ | $ABCD$ | $ABCD$ | $ABCD$ | $ABCD$ |

> *Six friends each know a secret. They can call each other by phone. In each call they exchange all the secrets they know. How many calls are needed for everyone to know all secrets?*[†]

We will consider the "friends" as *agents* in a multi-agent scenario, and then generalise the problem to the case of $n \geq 1$ agents. For now let us focus on protocols that are *successful* in the sense that they spread all secrets. If $n = 1$, no calls are needed. If $n = 2$, the two friends $a$ and $b$ need to make only one phone call, which we denote by $ab$ ('$a$ calls $b$'). For $n = 3$, the call sequence $ab, bc, ca$ will do.

Before we continue, let us briefly clarify some terminological assumptions. We see a secret as a propositional variable such that 'knowing the secret' means knowing the truth value of that variable. 'Agent $a$ knows secret $A$' means that agent $a$ knows whether $A$, i.e., agent $a$ knows that $A$ is false or agent $a$ knows that $A$ is true. We represent by $ab$ a call from $a$ to $b$. The informative consequences of a call (i.e., which secrets are exchanged) are independent from who initiates a call, so in that sense a call $ab$ is the same as a call $ba$. But, as we will see later, for the generating protocols the order makes a difference. Furthermore, we prefer to talk about secrets and not about propositions, propositional variables, or facts. Knowing a proposition or a fact tends to mean that you know *that* it is true, whereas knowing a secret tends to mean that you know whether it is true. It is in this sense that we wish to consider the knowledge that the agents have about the secrets in the gossip scenario described.

Consider a scenario with $n = 4$ agents, $a, b, c, d$, who hold, respectively, secrets $A, B, C, D$. The four calls $ab; cd; ac; bd$ distribute all secrets, and the underlying protocol, of which this call sequence is an execution, is as follows.

**Four-agent Protocol** Any two agents make the first call; the second call is then between the remaining two agents; the third call is then between an agent who made the first call and an agent who made the second call; and the fourth call is between the two who were not chosen in the third call. The distribution of secrets given the four calls is as shown in Table 1.1. (The rows list the distribution of secrets after a particular call took place.) We can also show that no other protocol solves this in four calls, and that

[†]Presented as a puzzle at the 1999 Nationale Wetenschapsquiz (National Science Competition), Netherlands.

less than four calls is insufficient to spread all secrets among all the agents. First, in an execution of any other protocol, one of the first callers will also make the second call. So, it has to start like this:

| | $a$ | $b$ | $c$ | $d$ |
|---|---|---|---|---|
| | $A$ | $B$ | $C$ | $D$ |
| $ab$ | $AB$ | $AB$ | $C$ | $D$ |
| $ac$ | $ABC$ | $AB$ | $ABC$ | $D$ |
| | $\ldots$ | | | |

How will this continue? For the third call, let us distinguish between the case that agent $d$ is not involved and the case that it is involved. If agent $d$ is not involved, then another call $ac$ does not result in more information. Let the third call be $ab$ (other possible calls at this point are $ba$, $bc$ or $cb$, but these cases will give rise to the same distribution of secrets as the $ab$ call, so suffice it here to consider only the $ab$ call). The distribution of secrets after the third call is then as follows:

| | $a$ | $b$ | $c$ | $d$ |
|---|---|---|---|---|
| | $A$ | $B$ | $C$ | $D$ |
| $ab$ | $AB$ | $AB$ | $C$ | $D$ |
| $ac$ | $ABC$ | $AB$ | $ABC$ | $D$ |
| $ab$ | $ABC$ | $ABC$ | $ABC$ | $D$ |
| | $\ldots$ | | | |

Three more calls are now required in order for all agents to know all secrets: for example $da; db; dc$ or $ad; ab; ac$. This makes *six* calls altogether.

But if the third call involves agent $d$, there also will always remain two agents who do not know $d$ yet. Again, two or three further calls are needed, so that we need at least *five* calls altogether.

This also demonstrates that less than four calls is insufficient to distribute all secrets, because any execution starts with either $ab; ac$ (at least five calls to termination) or $ab; cd$ (at least four calls to termination), modulo a permutation of agents.

For $n = 3$ and $n = 4$, $2n-4$ calls are sufficient to distribute all the secrets. Let there now be $n > 4$ agents. Then this is also sufficient. Suppose the agents are $a, b, c, d, e, f, \ldots$.

**Fixed Schedule** Choose four agents from the set of agents $\mathsf{Ag}$, say $a, b, c, d$, and one of those four, say $a$. First, agent $a$ makes one call to each of the agents in $\mathsf{Ag} \setminus \{a, b, c, d\}$. Then, the Four-agent protocol is executed among agents $a, b, c, d$ (say, the calls $ab; cd; ac; bd$ are made). Finally, agent $a$ again makes one call to each of the agents in $\mathsf{Ag} \setminus \{a, b, c, d\}$.

This adds up to $(n-4)+4+(n-4) = 2n-4$ calls. For $n = 6$ we get $2n-4 = 8$ calls. Given six agents, $a, b, c, d, e, f$ whose respective secrets are $A, B, C, D, E, F$, an example

TABLE 1.2: **Information transitions for the sequence:** $ae; af; ab; cd; ac; bd; ae; af$**.**

| | $a$ | $b$ | $c$ | $d$ | $e$ | $f$ |
|---|---|---|---|---|---|---|
| | $A$ | $B$ | $C$ | $D$ | $E$ | $F$ |
| $ae$ | $AE$ | $B$ | $C$ | $D$ | $AE$ | $F$ |
| $af$ | $AEF$ | $B$ | $C$ | $D$ | $AE$ | $AEF$ |
| $ab$ | $ABEF$ | $ABEF$ | $C$ | $D$ | $AE$ | $AEF$ |
| $cd$ | $ABEF$ | $ABEF$ | $CD$ | $CD$ | $AE$ | $AEF$ |
| $ac$ | $ABCDEF$ | $AB$ | $ABCDEF$ | $CD$ | $AE$ | $AEF$ |
| $bd$ | $ABCDEF$ | $ABCDEF$ | $ABCDEF$ | $ABCDEF$ | $AE$ | $AEF$ |
| $ae$ | $ABCDEF$ | $ABCDEF$ | $ABCDEF$ | $ABCDEF$ | $ABCDEF$ | $AEF$ |
| $af$ | $ABCDEF$ | $ABCDEF$ | $ABCDEF$ | $ABCDEF$ | $ABCDEF$ | $ABCDEF$ |

TABLE 1.3: **Information transitions for the sequence:** $ab; cd; ef; ac; de; af; bd; ce$**.**

| | $a$ | $b$ | $c$ | $d$ | $e$ | $f$ |
|---|---|---|---|---|---|---|
| | $A$ | $B$ | $C$ | $D$ | $E$ | $F$ |
| $ab$ | $AB$ | $AB$ | $C$ | $D$ | $E$ | $F$ |
| $cd$ | $AB$ | $AB$ | $CD$ | $CD$ | $E$ | $F$ |
| $ef$ | $AB$ | $AB$ | $CD$ | $CD$ | $EF$ | $EF$ |
| $ac$ | $ABCD$ | $AB$ | $ABCD$ | $CD$ | $EF$ | $EF$ |
| $de$ | $ABCD$ | $AB$ | $ABCD$ | $CDEF$ | $CDEF$ | $EF$ |
| $af$ | $ABCDEF$ | $AB$ | $ABCD$ | $CDEF$ | $CDEF$ | $ABCDEF$ |
| $bd$ | $ABCDEF$ | $ABCDEF$ | $ABCD$ | $ABCDEF$ | $CDEF$ | $ABCDEF$ |
| $ce$ | $ABCDEF$ | $ABCDEF$ | $ABCDEF$ | $ABCDEF$ | $ABCDEF$ | $ABCDEF$ |

execution sequence[‡] of the Fixed Schedule for such a scenario is:

$$ae; af; ab; cd; ac; bd; ae; af \tag{1.1}$$

That is, agent $a$ starts by calling agent $e$ and then agent $f$, etc. We illustrate the distribution of secrets with the execution sequence, as shown in Table 1.2. After the protocol, all friends indeed know all secrets. Less than $2n-4$ calls are insufficient to distribute all secrets. This has been shown, for example, in [63] (see also [30, 32]). For $n > 4$, the Fixed Schedule is not the only protocol to distribute the secrets in $2n-4$ calls. For example, in any execution sequence of the Fixed Schedule some calls are made more than once (for the depicted $n = 6$ execution, these are $ae$ and $af$). The information transitions shown in Table 1.3 also achieves the distribution of all secrets over all agents but, in the corresponding execution sequence, all calls are different. Not all sequences of eight different calls distribute the secrets over all agents. For example, when we change the sixth call from $af$ into $bf$, agent $a$ will only know the secrets $A, B, C, D$ after those eight calls.

[‡]Throughout this thesis we will use the term “execution sequence" and “call sequence" interchangeably.

TABLE 1.4: **Information transitions for the sequence:** $ab; ac; ad; bc; bd; cd$**.**

| | $a$ | $b$ | $c$ | $d$ |
|---|---|---|---|---|
| | $A$ | $B$ | $C$ | $D$ |
| $ab$ | $AB$ | $AB$ | $C$ | $D$ |
| $ac$ | $ABC$ | $AB$ | $ABC$ | $CD$ |
| $ad$ | $ABCD$ | $AB$ | $ABC$ | $ABCD$ |
| $bc$ | $ABCD$ | $ABC$ | $ABC$ | $ABCD$ |
| $bd$ | $ABCD$ | $ABCD$ | $ABC$ | $ABCD$ |
| $cd$ | $ABCD$ | $ABCD$ | $ABCD$ | $ABCD$ |

Yet another protocol is obtained by imagining the agents lined up along a round-table, such that, starting with an agent, say agent $a_1$, each agent passes on its secrets to its neighbour in, say, the clockwise direction, until we have almost come full circle twice (after $n-1$ calls, both $a_n$ and $a_{n-1}$ know all secrets; it takes only $n-2$ calls to pass those on to $a_1, a_2, \ldots, a_{n-2}$). This gives rise to $2n-3$ calls, only one more than the minimum of $2n-4$. An example execution sequence is shown as follows:

$$a_1a_2; a_2a_3; \ldots; a_{n-1}a_n; a_na_1; a_1a_2, a_2a_3; \ldots; a_{n-2}a_{n-1} \tag{1.2}$$

**Maximum number of calls** If gossip is the goal, then prolonging gossip is better. As long as two agents who call each other still exchange all the secrets that they know and at least one of them learns something new from the call, what is the *maximum* number of calls to distribute all secrets?

The maximum number of calls to distribute all secrets is $\binom{n}{2} = \frac{n\cdot(n-1)}{2}$ (see Proposition 6.12 later). This is also the maximum number of different calls among $n$ agents. For six agents $a, b, c, d, e, f$ the following calls can be made such that in every call at least one agent learns one secret—for convenience we generate the execution sequence in lexicographic order.

$$ab; ac; ad; ae; af; bc; bd; be; bf; cd; ce; cf; de; df; ef$$

For four agents we get:

$$ab; ac; ad; bc; bd; cd$$

We give the detailed distribution of secrets for four agents as shown in Table 1.4.

## 1.2 Overview of the Literature on the Gossip Problem

Earlier in the chapter, we began by presenting the gossip problem as a puzzle challenge. Let us now briefly identify some of the various other formulations of this problem in the early literature of the problem, based on the combinatorial survey [30]. Our aim is to present the foundation on which we build the idea of epistemic gossip protocols.

The gossip problem has been viewed from several perspectives. For example, Hajnal et al. [27] presented the problem in a similar form as we did earlier in the chapter, but then in terms of "ladies" and "scandals". Bavelas [8] formulated the gossip problem in terms of "playing cards", where the members of a group have the goal of producing the highest-ranking poker hand that can be made by selecting one card from each subject. Shimbel [60] expressed the problem in the setting of a network of nodes, where each node has the goal of learning the neighbours of every other node in the network, whereas Landau [41] treated the problem in terms of coloured marbles - each individual in a group is given a set of such marbles, but one colour is common among all the sets of marbles. The goal is for the group to learn the common colour by means of pairwise messaging. In Cederbaum [14], nodes in a communication graph are initially given a unique value $N$ from the set $\{1, \ldots, n\}$. At each round $r$ of gossiping, the node labelled $N = r$ sends all the secrets it currently possesses to all its neighbours. The solution involves producing a sequence of node relabellings that spreads the secret of each node to all other nodes.

The gossip problem has been approached with various motivations. For example, whereas the work by Shimbel [60] focused on a communication network (e.g., a telephone network), the work by Bavelas [8] sought an understanding of how graph structures influence group tasks in social networks. In [12, 35, 46], the gossip problem surfaces as an application in developing algorithms for various numeric computations. Miklos et al. [46], for example, describes an 'addition game'. In this game, each of the $n$ players initially holds a number whose value is one. In each pairwise communicative interaction (or telephone call) between the agents, the value of the number held by each calling player is updated to the sum of the values of the calling pair. Following this rule, the aim of this game is to make the value of the number of each player equal to $n$.

However, there are common questions running through the various formulations and motivations around various studies of the gossip problem, namely: how many communicative interactions are necessary to successfully reach the goal of gossiping? (The goal of gossiping is, in general, that every agent knows the piece of information initially held by each other.) How does the network structure affect the problem? Which communication sequences solve the problem, and what are the characteristics of various such sequences? Which general schemes or procedures yield solutions to the problem?

The minimum number of pairwise communicative interactions[§] needed for successful gossiping was proved in [13, 27, 61, 63]. They all showed that for $n \geq 4$, this minimum is given by the expression $2n-4$. Farley and Proskurowski [23] proved that for a line graph, the minimum number of pairwise communicative interactions for successful gossiping is $2n - 3$. Harary and Schwenk [28] showed that if the underlying communication graph

[§]Note that in the discussions in this chapter, and throughout this thesis, when we refer to an underlying communication graph or network graph, we also assume that the graph is undirected, unless otherwise stated. It is assumed that the direction of an edge on a graph indicates which of the two adjoining nodes can disclose their secrets during a pairwise communicative interaction or phone call (only a node with an outgoing edge can disclose the secrets they know during such a communicative interaction). As such, when we assume that a graph is undirected we indicate that in each pairwise communicative interaction, the communicating pair tell each other all the secrets they know.

is a tree then this minimum is given by $2n - 3$, for $n \geq 2$. Also, [28] and [26] showed that if a connected communication graph contains a 4-cycle, then the minimum number of pairwise communicative interactions is $2n - 4$, for $n \geq 4$. However, Bumby [13] and Kleitman [36] showed that for any connected but incomplete communication graph without a 4-cycle, this minimum is $2n-3$. Harary and Schwenk [28] also showed that for a strongly connected directed communication graph with $n$ nodes, the minimum number of calls required for successful gossiping is $2n - 2$. This result corresponds with the case where there is only a one-way communication between the nodes in the communication graph, as in *text messaging*.

Instead of consecutive telephone calls wherein all secrets are exchanged between both parties, several calls between pairs of agents might as well take place at the same time, to speed up the exchange of information. In that case, what is the minimum number of rounds to communicate all secrets between $n$ agents? The answer is given in [37] as: $\lceil log_2 n \rceil$ for $n$ even, and $\lceil log_2 n \rceil + 1$ for $n$ odd.

Motivated by the goal of minimising various redundancies in communicative interactions in a gossip execution sequence, some researchers began to study general procedures that give rise to execution sequences with specific properties. For example, Cot [18] found that for $n = 4$ and $n = 8$ nodes, it is possible to gossip successfully in the minimum number $(2n-4)$ of pairwise interactions, and still have that the pairwise interactions are between nodes that do not yet know all the secrets. West [69] showed that for all even $n \geq 4$ agents, there are gossip procedures that produce successful execution sequences in which no agent learns its own unique secret from another agent in any communicative interaction. West also gave some of the graph-theoretic properties of the graphs that allow for such procedures. Seress [57] proved that for all even $n$ (except $6, 10, 14$ and $18$), there are successful execution sequences in which no agent learns a secret that it already knows. Still under the restriction that no agent learns a secret that it already knows, [59] showed that where $n$ is divisible by 4, and $n \geq 8$, the number of communicative interactions required for successful gossiping is $\frac{9}{4}n - 6$. However, in [58], Seress showed that there are no successful execution sequences under this restriction if the underlying communication graph allows only one-way communicative interactions, as in directed graphs. Finally, other than the purpose of minimising redundancy in communicative interactions among agents, Harary and Schwenk [29], for example, focus on how well the underlying communication graph is utilised.

The very close precursors to our work are those that study gossiping under randomly staged communicative interactions. For one-way communication graphs, Landau [41] determined the average number of communicative interactions for successful gossiping if a pair of agents is randomly chosen to communicate in each round. Note that several pairwise communicative interactions may be possible in each round, but only one of them is randomly chosen. Moon [47] studied the same problem using undirected (two-way) communication graphs. Again, several pairwise communicative interactions may be possible in each round but only one of them is randomly chosen. In each pairwise

communicative interaction, the pair of agents exchange all the secrets they currently know. Under this two way communication graph, Boyd and Steele [11] showed that the average number $A(n)$ of pairwise communicative interactions needed for successful gossiping among $n$ agents is given by:

$$A(n) = \frac{3}{2} n \; ln \; n + O(n(ln \; n)^{0.5})$$

## 1.3 Epistemic Gossip Protocols

In the approaches discussed so far, there is a global scheduler that orders the call actions among the agents. The Fixed Schedule, for example, begins by coordinating the agents: a designated group of four agents is chosen (in our examples, the agents in this group are $a, b, c, d$), then one agent from the designated group is further designated to call the non-designated agents (in our example this agent is $a$). So the scheduler, while adhering to the protocol, assigns to each agent the calls to be made. In the example execution sequence of the Fixed Schedule given earlier in this chapter, the agent named $a$ is assigned to make the first call, and to make this call to another agent named $b$; then, the agent named $c$ is assigned to call the agent named $d$. And so on. This kind of pre-coordination would be natural for instance if the agents are likened to a subset of a cohort of students who have common knowledge that some specific exam result will be made available to each of them individually (so the secret of agent $a$ is either '$a$ passed' or '$a$ failed'), and the students can lookup a plan on how to communicate their results to each other, before the exam results are initially revealed to each student. However, a better way to define a protocol seems to be, for example, that any two agents can make the first call, and then any two other agents can make the second call. So it does not have to be between a predetermined set of four individuals. Furthermore, often such pre-coordination is not possible. Suppose all students of the said cohort receive an unexpected invitation for a party. The students may be curious to find out about each other whether they will accept, in which case they will have to make phone calls based on the knowledge, or better ignorance, they have about the secrets of others. Since in such a distributed protocol several agents may decide to initiate a call at the same time, we assume the presence of an *arbiter* who breaks a tie in such cases.

Still considering the Fixed Schedule, the question can then be asked whether the protocol can be rephrased such that these scheduling decisions can be made by the agents themselves, at execution time, and based on what they know. One can still imagine the first two callers being determined randomly. One of them is simply the agent getting through before the others, in making a call, and the recipient of that call can be any other agent. All agents only know their own secret initially. But for the second call we have a problem. Now there are two agents who know two secrets, and the remaining agents only know one secret. In other words, they have different knowledge. We may pick any agent who only knows one secret to initiate a call, this choice is knowledge-based (and anyone fulfilling the condition can be chosen), and this rules out those who made the call

in the first round. In our attempt to generalise the Fixed Schedule, this agent now has to call another agent who only knows one secret. But the agent initiating that second call cannot choose such a one-secret-only agent *based on its knowledge*. If agent $c$ initiates the second call, it has no reason to prefer agent $d$ over any other agent, if agent $c$ were ignorant about who made the first call. It seems not unreasonable to assume that agent $c$ only knows that it was not involved herself in that first call. That means that, from the point of view of agent $c$, the first call could have been between agent $a$ and agent $b$, or between agent $a$ and agent $d$, or between agent $b$ and agent $d$. (And from each pair, either agent initiating the call.) Agent $c$ does not know which one really happened. We can also say that agent $c$ cannot distinguish different *histories* of calls, as in [50, 51].

In this thesis we study such epistemic gossip protocols wherein an agent calls another agent based on its knowledge (or ignorance) only, and when more than one call is enabled simultaneously by such criterion, we assume an arbiter which randomly selects one of the possible calls. Hence, the agents are enriched with full autonomy within the gossip scenario. We will describe a number of epistemic gossip protocols and analyse their performance on the gossip problem.

### 1.3.1 Dynamic Epistemic Logic and Epistemic Planning

The planning problem in the field of artificial intelligence can be described as follows: given an *initial state* of the world, and a description of a *desired state*, how can the initial state be changed into the desired state [56, Section 10.1]. Typically, the set of actions that can be performed in the scenario are also given. The change from the initial state to the desired state may be accomplished by a sequence of such allowable actions starting from the initial state, but sometimes such desired states may not be reachable by any sequence of the allowable actions. Such a sequence of allowable actions is a plan. A *successful* plan gives a way to change an initial situation into a desired situation, thus yielding a solution to the planning problem.

Another way of giving the solution to a planning problem is to describe a high level scheme, procedure or rule which gives rise to a set of plans or programs. We refer to such high level scheme as a protocol. Such protocols may also generate plans that do not give rise to the desired state of the system. Hence another consideration within the planning problem is whether an agent can know for sure that a current plan will lead to a desired state, or whether a protocol guarantees that the desired state will be reached. The planning problem becomes more involved when it is extended to scenarios with multiple agents, or scenarios wherein the set of allowable actions differ for each of the agents, or scenarios wherein the allowable actions may be performed by any one of the agents or jointly by some subset of the given agents.

In epistemic planning, the *initial state* corresponds to an initial state of knowledge (or epistemic state) of an agent or a group of agents, whereas the desired state corresponds to a desired state of knowledge of an agent or a group of agents. The solution to the epistemic planning problem can likewise be described as a plan consisting of a sequence of

allowable epistemic actions performed by the agents; and on the other hand, an epistemic protocol consists of knowledge-based or epistemic rules that give rise to a set of epistemic plans. Such epistemic rules may be expressed in terms of the knowledge of various agents, and agents may act synchronously (at fixed and commonly known time intervals) or asynchronously, at individually convenient time intervals (which may make the desired state of knowledge more difficult to accomplish).

The planning problem is related to the *synthesis problem*, which, according to Church [16, 17], is as follows:

> *Given a requirement which a circuit is to satisfy, we may suppose the requirement is expressed in some suitable logistic system which is an extension of restricted recursive arithmetic. The synthesis problem is then to find recursion equivalences representing a circuit that satisfies the given requirement (or alternatively, to determine that there is no such circuit).*

We consider the synthesis problem from the point of view of program synthesis in computer science and artificial intelligence. The synthesis problem has two components, namely, the decision component and the synthesis component. The decision component asks whether there exists a program which changes the input state of the given system into the desired state, whereas the synthesis component describes a procedure by which such a program is synthesised when it is the case that such a program exists. Formal logics, especially temporal logics, are typically used to specify a program in terms of its inputs, its outputs and its behaviour with respect to the inputs and outputs of the system. Hence the question arises for each such specification whether there is a program that satisfies the specification.

Given that $x$ is the variable for the input to a program, and $y$ is the variable for the output of the program. Let $\varphi(x, y)$ be a linear time temporal specification for a single agent system. Pnueli and Rosner [53] gave and proved the condition for the existence of a program that satisfies the specification $\varphi(x, y)$. They gave an algorithm for synthesising the program if it exists and showed that whether such a program exists is decidable if $x$ and $y$ range over finite domains. Kupferman and Vardi [39] solve the single agent synthesis problem for the temporal logics CTL and CTL* specifications [20, 21, 40]. For program synthesis with incomplete information (part of the input is unavailable), and for temporal specifications for a single agent system, Meyden and Vardi [65] used a knowledge-based or epistemic approach to express the uncertainty of the agent about its environment. Thus they solved the single agent synthesis problem for specifications given in temporal epistemic logic. For specifications given in multiagent temporal epistemic logic, Pnueli and Rosner [54] prove that for two agents, synthesis under incomplete information is already undecidable. Nevertheless, there are some special cases of multi agent temporal epistemic specifications that are decidable. For example, Pnueli and Rosner [54] prove that in the setting where the observational powers of the agents form a total order, then program synthesis from the temporal epistemic specification of such a setting is decidable.

When we consider input and output states that are a description of the knowledge (and possibly higher-order knowledge) of agents in a system, and consider actions that can change the knowledge of such agents (epistemic actions), then such input and output states are regarded as epistemic, and a *program or plan* (a sequence of actions) that changes the input or initial state of the system to the desired or output state is regarded as an *epistemic plan* or *epistemic program*.

For a multiagent system specification in dynamic epistemic logic [66, 67], Bolander and Andersen [10], show that single agent epistemic planning is decidable whereas the general case of multiagent epistemic planning is undecidable. Löwe et al. [44] describe a tractable fragment of epistemic planning which is due to the restriction of allowable actions to "almost-mutually-exclusive" actions with only propositional preconditions, and having transitive accessibility relations. They also show that such a fragment is interesting by using it to formalise and reason about change of knowledge of agents due to actions performed in a multiagent video game example. Parikh and Krasucki [49] characterise the conditions under which a sequence of communications that realise a specified level of knowledge can be synthesised, for a distributed system, where a level of knowledge is a formula of the form $K_iK_j \dots A$, and $\{i, j, \dots\}$ is the set of processes (agents) in the distributed system. They also provide an algorithm to synthesise such a protocol where it exists. Chandy and Misra [15] give a characterisation of the number of communications required by processes to know specific facts about a distributed system.

Let $\varphi$ be a dynamic epistemic logic formula that describes what is true about an initial situation; let $\varphi'$ be a dynamic epistemic logic formula that describes what is true about an action occurring at the initial situation; and let $\varphi''$ be a dynamic epistemic logic formulae that describes what is true about the resulting situation due to the action occurring at the initial situation. Aucher [6, 7] provide the following axiomatisations: (a) an axiomatisation of all the information that is true in the resulting situation given $\varphi$ and $\varphi'$, (b) an axiomatisation of all the information that is true in the initial situation given $\varphi'$ and $\varphi''$, and (c) an axiomatisation of all the information that is true about the action taking place at the initial situation, given $\varphi$ and $\varphi''$. For a finite set of atomic actions, Aucher [7] described a way to synthesise the action that changes a given initial situation to a situation satisfying a dynamic epistemic logic formula that expresses what is true in the resulting situation.

Our approach to the gossip problem can be considered as a special case of the planning problem, namely, epistemic planning. Based on Bolander et al. [10] in which the epistemic planning problem is defined as a special case of the classical planning problem, an epistemic planning problem is a tuple $EP = (S, Act, \gamma, s_0, \varphi_g)$, where: $S$ is a finite set of epistemic situations; $s_0 \in S$ is the initial situation; $Act$ is a finite set of epistemic actions; $\gamma$ is a function that takes a situation $s \in S$ and an action $a \in Act$, and returns a situation that is a result of executing the action $a$ at the situation $s$. Note that $\gamma$ is undefined if the action $a$ cannot be executed in the situation $s$; finally, $\varphi_g$ is an epistemic goal. The solution to the epistemic planning problem is then a sequence $\sigma$ of actions

$\alpha \in Act$, such that the execution of $\sigma$ at $s_0$ gives rise to a situation that satisfies the goal formula $\varphi_g$.

Likewise, the gossip problem $GP = (\mathcal{S}, \mathcal{C}, F, M^{init}, \varphi_g^e)$ can defined as a special case of the epistemic planning problem as follows:

- $\mathcal{S}$ is a finite set of gossip situations,
- $\mathcal{C} = \{ab \mid a \neq b \in \mathsf{Ag}\}$ is a set of call actions,
- $F$ is a function that takes a gossip situation $M \in \mathcal{S}$ and a call $ab \in \mathcal{C}$ and returns a gossip situation that is a result of executing the call action $ab$ at $M$,
- $M^{init} \in \mathcal{S}$ is the initial gossip situation,
- $\varphi_g^e$ is an epistemic formula which expresses the goal of gossip, which typically is, that every agent knows the secret of every other agent in the set $\mathsf{Ag}$ of agents.

And, analogous to epistemic planning, the solution to the gossip problem can also be expressed as a sequence $\sigma_e$ of calls $c \in \mathcal{C}$ such that the execution of $\sigma_e$ at the initial situation $M^{init}$ gives rise to a situation that satisfies the goal formula $\varphi_g^e$. The reader is referred to Chapter 3 for details of the elements of the gossip problem.

## 1.4 Aim of the Thesis

In this thesis we argue that distributed epistemic gossip protocols[¶] can provide a solution for the gossip problem of successfully spreading the information in a network of agents, while enriching the agents with autonomy.

### 1.4.1 Research Questions

The thesis is based on four main research questions, as follows.

**Research Question 1:** *Is it possible to describe epistemic gossip protocols for the successful spreading of information in a network of autonomous agents?*

We provide the answer to this research question in Chapter 3, Chapter 5 and Chapter 6 of the thesis. In Chapter 3 we describe a number of epistemic gossip protocols. We study the properties of these protocols from an empirical point of view (in Chapter 5) and from an analytical point of view (In Chapter 6).

[¶]Recall that throughout this thesis, we will use the terms 'distributed epistemic gossip protocol' and 'epistemic gossip protocol' interchangeably, since we always assume that each agent executes its own protocol.

**Research Question 2:** *Is it possible to create, and use, some formalism based on dynamic epistemic logic to specify, model, analyse and verify epistemic gossip protocols?*

We provide the answer to this question mainly in Chapter 3 of the thesis, where we present a formalism based on dynamic epistemic logic for reasoning about epistemic properties of agents in a gossip scenario. We extend our formalism in Chapter 4, a difference being that we then interpret epistemic formulas on a more abstract structure than that presented in Chapter 3. In both Chapter 3 and Chapter 6, we use our formalism to study the logical properties of epistemic gossip protocols.

**Research Question 3:** *Is it possible to create a software framework to automate the empirical analysis of epistemic gossip protocols, given a specification of such a protocol?*

In Chapter 4, we build upon the work in Chapter 3 by implementing a software framework to automate the empirical analysis of epistemic gossip protocols. In Chapter 5, we present and discuss the empirical results obtained from the use of our software tool in the analysis of our epistemic gossip protocols.

**Research Question 4:** *How does various network connectivity constraints affect the properties of epistemic gossip protocols?*

We do an empirical study (in Chapter 5), and an analytical study (in Chapter 6), of how various epistemic gossip protocols perform on various types of network graphs.

### 1.4.2 Methodology

The research methodology that we employ in answering the research questions posed in the preceding subsection, is as follows.

**Formal Logic:** We create a formalism based on dynamic epistemic logic, by which to specify and model epistemic gossip protocols.

**Protocol Specification and Analysis:** We specify a number of epistemic gossip gossip protocols using our logical formalism, and we prove a number of properties of these protocols on various types of network graphs.

**Framework Implementation and Experimentation:** We implement a software framework to be used in the automated analysis of epistemic gossip protocols. Using our software tool, we perform empirical studies on a number of such protocols, with various network graph constraints. And, we interpret the results of our experiments to indicate the performance of these protocols in themselves and in comparison with each other.

### 1.4.3 Overview of the Thesis

In this chapter we introduced the gossip problem, gave an overview of the literature on the problem, introduced an epistemic approach to the solution of the gossip problem, and presented our research questions and methodology for the thesis.

In Chapter 2 we provide technical background to the work that will be presented in this thesis. The main contributions of this thesis are then presented in the next four chapters, from Chapter 3 through Chapter 6.

In Chapter 3 we present some epistemic gossip protocols. We introduce a formal logic for the formalisation of epistemic gossip protocols. We discuss and prove some of the properties of our logical formalism, and we use this formalism to describe our epistemic gossip protocols.

In Chapter 4 we introduce a framework for automated empirical analysis of epistemic gossip protocols. Within this framework, we introduce a high level programming language for describing epistemic gossip protocols, and a program interpreter for this high level language. An epistemic gossip protocol is then described in form of a program, and translated into an abstract model of the gossip protocol, from which some key performance characteristics of the protocol are determined.

In Chapter 5 we use the framework introduced in Chapter 4 to describe and analyse the epistemic gossip protocols we described in Chapter 3. We then provide results of experiments on the performance of these protocols.

In Chapter 6 we describe some of the properties of epistemic gossip protocols for various network topologies, and then prove some of the properties of the epistemic gossip protocols introduced in Chapter 3 for various network types of network graph.

Finally, Chapter 7 concludes the thesis and provides some ideas for further research.

**Sources of Materials** Chapters 3, 4, 5 and 6 are based on collaborations with Prof. Wiebe van der Hoek, Dr. Davide Grossi and Prof. Hans van Ditmarsch. The core of the work in Chapter 3 was published in the proceedings of European Conference on Artificial Intelligence (ECAI) 2014 [3]. Some part of the work in Chapter 3 was also published as a contribution to a book in honour of Rohit Parikh [5]. The core of the work in Chapter 4, and part of Chapter 5, was published in the proceedings of European Conference on Multi-Agent Systems (EUMAS) 2014 [4]. The contributions and collaborations of my co-authors are highly appreciated.

# Chapter 2

# Background

In this thesis, we formalise epistemic gossip protocols using dynamic epistemic logic (DEL). Our objective in this chapter is twofold. First, to provide the reader with some of the rudiments of dynamic epistemic logic which we build upon in other chapters of this work. The second objective is to specify some of the notation and assumptions we use throughout this thesis. The material in this chapter is mainly based on [45] and [67], to which the reader is referred for more details and examples.

Dynamic epistemic logic is a logical formalism which can be used to describe and reason about the change of information in agents as a result of communication. We begin our presentation with a basic epistemic logic (which has no dynamic operators and as such does not describe information (or knowledge) change). The aim is to use this basic logic as a vehicle to convey the background theory on which our later presentation of DEL is based.

## 2.1 Basic Epistemic Logic

**Definition 2.1** (Basic Epistemic Language, $\mathcal{L}$)**.** Given a set of atomic propositions $\mathsf{P}$, and a set $\mathsf{Ag}$ of agents, the language $\mathcal{L}$ of multi-agent epistemic logic is defined by the following BNF grammar:

$$\varphi \quad ::= \quad A \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid K_a\varphi$$

where $A \in \mathsf{P}$ and $a \in \mathsf{Ag}$.

**Notation.** We read $K_a\varphi$ as "agent $a$ knows that $\varphi$". Where we use the symbols $a, b, c, \ldots$ for agent names, we will use the symbols $A, B, C, \ldots$ for propositional atoms. And where we use $a_i, a_j, a_k, \ldots$ for agent names, we will use the symbols $A_i, A_j, A_k, \ldots$ for propositional atoms. Throughout this thesis we will often talk about the *unique secret* of an agent, that is, the secret that an agent possesses *ab initio*. Such secret is treated as a propositional atom. We will use an agent name and the secret of such agent in tandem, as follows: the name of an agent is $a$ if and only if the unique secret of agent

$a$ is the propositional atom $A$. More elaborately, the agent names are $a, b, c, d, e, f, \ldots$ if and only if the unique secret of each of these agents, respectively, is $A, B, C, D, E, F, \ldots$. Likewise, where the agent names are $a_i, a_j, a_k, a_l, a_m, a_n, \ldots$, the respective unique secret of each agent is $A_i, A_j, A_k, A_l, A_m, A_n, \ldots$.

We will also assume some conventional abbreviations such as:

$$\begin{array}{rcc} \top & = & A \wedge \neg A \\ \bot & = & \neg\top \\ (\varphi \vee \psi) & = & \neg(\neg\varphi \wedge \neg\psi) \\ (\varphi \rightarrow \psi) & = & \neg\varphi \vee \psi \\ (\varphi \leftrightarrow \psi) & = & (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi) \end{array}$$

Regarding other notation used throughout the thesis, we may sometimes leave out parenthesis when writing formulas of a formal language, if doing so does not lead to confusion. Also, especially in the proofs presented, we may sometimes use $\Longrightarrow$, $\Longleftrightarrow$, $\forall$ and $\exists$ as abbreviations for the meta-logical expressions 'implies', 'is equivalent to', 'for all' and 'there exists', respectively. We may also use the convenient word 'iff' for 'if and only if', and, where $a \in \mathsf{Ag}$ and $\varphi$ is a logical formula, we will sometimes write $\neg K_a \neg\varphi$ as $\hat{K}_a\varphi$.

We now give examples of the use of formulas of the language $\mathcal{L}$, as follows.

**Example 2.1.** *Consider a gossip scenario with three agents $a, b, c$. The unique secret of agent $a$ is the propositional atom $A$; the unique secret of agent $b$ is the propositional atom $B$; and the unique secret of agent $c$ is the propositional atom $C$. At the start, every agent knows only its own unique secret. That is, $(K_aA \vee K_a\neg A) \wedge (K_bB \vee K_b\neg B) \wedge (K_cC \vee K_c\neg C)$ holds, but we also have that agent $a$, for example, does not know the secret of agent $b$ and agent $c$, that is, $\neg K_aB \wedge \neg K_a\neg B \wedge \neg K_aC \wedge \neg K_a\neg C$. Moreover, agent $a$ knows that agent $b$ knows only secret $B$, and that agent $c$ knows only secret $C$ (each agent knows of the unique secret of each of the other agents, and knows that each of the other agents knows only their unique secret). So we have, for example, $K_a(K_bB \vee K_b\neg B) \wedge K_a\neg(K_bA \vee K_b\neg A) \wedge K_a\neg(K_bC \vee K_b\neg C)$. Let the situation described in this example be called the* initial gossip situation. *Later in Example 2.2, we will give more epistemic properties of this initial gossip situation.*

### 2.1.1 Semantics of the language $\mathcal{L}$

Let us now address the question of how to give a precise meaning to sentences of the language $\mathcal{L}$. Following the usual practice in modal logics, we express the meaning of the sentences of the language $\mathcal{L}$ in terms of the elements of a mathematical structure known as a Kripke model.

**Definition 2.2** (Kripke Model)**.** Let $\mathsf{P}$ be a countable set of atomic propositions, and let $\mathsf{Ag}$ be a finite set of agents. A Kripke model is a triple, $M = (S, R, V)$, where $S$, the domain of $M$, is a set of states; and $R : \mathsf{Ag} \rightarrow S \times S$ is a function that assigns an

accessibility relation to $a \in \mathsf{Ag}$; and $V : S \to \mathsf{P} \to \{0, 1\}$ assigns a truth value ('0' for *false*, and '1' for *true*) to each atom in any given state of $M$.

Given a Kripke model $M = (S, R, V)$, let $a$ be an arbitrary agent from $\mathsf{Ag}$, then some of the classes of models in epistemic logic are as follows:

1. The class of all Kripke models.
2. The class of Kripke models where the accessibility relation $R(a)$ is *serial*: the accessibility relation $R(a)$ is serial if for all $s \in S$ there is a $t \in S$ such that $(s, t) \in R(a)$.
3. The class of Kripke models where the accessibility relation $R(a)$ is *reflexive*: the accessibility relation $R(a)$ is reflexive if for all $s \in S$, $(s, s) \in R(a)$.
4. The class of Kripke models where the accessibility relation $R(a)$ is *symmetrical*: the accessibility relation $R(a)$ is symmetrical if for all $s, t \in S$, if $(s, t) \in R(a)$ then $(t, s) \in R(a)$.
5. The class of Kripke models where the accessibility relation $R(a)$ is *transitive*: the accessibility relation $R(a)$ is transitive if for all $s, t, u \in S$, if $(s, t) \in R(a)$ and $(t, u) \in R(a)$ then $(s, u) \in R(a)$.
6. The class of Kripke models where the accessibility relation $R(a)$ is *Euclidean*: the accessibility relation $R(a)$ is Euclidean if for all $s, t, u \in S$, if $(s, t) \in R(a)$ and $(s, u) \in R(a)$ then $(t, u) \in R(a)$.
7. The class of Kripke models where the accessibility relation $R(a)$ is an *equivalence relation*: the accessibility relation $R(a)$ is an equivalence relation if $R(a)$ is reflexive, transitive and symmetrical. The class of Kripke models with equivalence relations is denoted by $S5$.

**Definition 2.3** (Epistemic model)**.** An epistemic model $M$ is a tuple $M = (S, \sim, V)$ such that:

- $S$ is a non-empty set of possible worlds,
- $\sim : \mathsf{Ag} \to \mathcal{P}(S \times S)$ assigns an equivalence relation to each agent,
- $V : S \to \mathsf{P} \to \{0, 1\}$ is a valuation for each $s \in S$.

**Notation** As in the definition of Kripke model (Definition 2.2), assigning the value '1' to an atomic proposition can be thought of as assigning the value *true* to it. Likewise, assigning the value '0' to an atomic proposition can be thought of as assigning the value *false* to it. If $M = (S, \sim, V)$, rather than $s \in S$, we will also write $s \in M$. For $\sim(a)$ we will write $\sim_a$, and for $V(s)$ we will write $V_s$. A *pointed epistemic model* is a pair $(M, s)$ where $s \in M$. We will also consider multi-pointed epistemic models $(M, S')$, where $S' \subseteq S$.

Note that a Kripke model $M = (S, R, V)$ is an epistemic model if for all $a \in \mathsf{Ag}$, $R(a)$ is an equivalence relation. Notice that since $R(a)$ is an equivalence relation for the case of an epistemic model, for all $a \in \mathsf{Ag}$, instead of writing $M = (S, R, V)$ we write $M = (S, \sim, V)$, and whenever any pair $s, t \in S$ is such that $(s, t) \in \sim_a$, we write $s \sim_a t$ instead.

**Definition 2.4** (Interpretation of Formulas of $\mathcal{L}$)**.** We will interpret epistemic formulas on pairs $(M, s)$ consisting of an epistemic model $M = (S, \sim, V)$ and a state $s \in S$. Properly speaking, $(M, s)$ is called an *epistemic state* but we will sometimes refer to it simply as a *state*. Note also that sometimes we write $M, s$ instead of $(M, s)$. We write $M, s \models \varphi$ for 'a formula $\varphi$ holds in $(M, s)$', and instead of 'not $M, s \models \varphi$' we write '$M, s \not\models \varphi$'. The truth condition for such claims is as follows:

$$
\begin{array}{lll}
M, s \models A & \text{iff} & V_s(A) = 1 \\
M, s \models (\varphi \wedge \psi) & \text{iff} & M, s \models \varphi \text{ and } M, s \models \psi \\
M, s \models \neg\varphi & \text{iff} & M, s \not\models \varphi \\
M, s \models K_a\varphi & \text{iff} & \text{for all } t \text{ such that } s \sim_a t, \quad M, t \models \varphi
\end{array}
$$

If $M, s \models \varphi$ for all $s \in T \subseteq S$, we write $M, T \models \varphi$. If a formula $\varphi$ holds in $(M, s)$ for all $s \in S$, where $S$ is the domain of $M$, we write $M \models \varphi$. Alternatively, we say that such a formula $\varphi$ is valid in $M$. If a formula $\varphi$ is valid in all $M \in \mathcal{C}$ where $\mathcal{C}$ is a subclass of the class $\mathcal{K}$ of all Kripke models, then we write $\mathcal{C} \models \varphi$. Alternatively, we say that such formula $\varphi$ is valid in the class of models $\mathcal{C}$ or that $\varphi$ is $\mathcal{C}-$valid. If a formula $\varphi$ is valid in all $M \in \mathcal{K}$ then we write $\mathcal{K} \models \varphi$ or we write $\models \varphi$. On the other hand, we use $\not\models$ to deny any of such claims. For example, when we write that $M \not\models \varphi$, we mean to say that it is not the case that the formula $\varphi$ holds in all $s \in S$. And this also implies that there is some $s \in S$ such that $M, s \not\models \varphi$. Moreover, if there is some $s \in S$ such that $M, s \models \varphi$, we say that the formula $\varphi$ is satisfied in $(M, s)$. We often use the abbreviation $Kw_aB$ for the formula $K_aB \vee K_a\neg B$.

In drawing epistemic models we will represent the equivalence relations using undirected lines. Also, for the sake of visual clarity, we will leave out the reflexive and transitive accessibility links in our graphical representations of such models. To give an example, the drawing shown in Figure 2.1 is a drawing of an epistemic model of the initial gossip situation described in Example 2.1. Note that each of the states is represented by a three-digit binary number. The digit at the most significant bit represents the value of the secret of agent $a$, whereas the middle bit and the least significant bit represent the value of the secret of agent $b$ and $c$, respectively. The bit value '1' means that the value of the corresponding propositional atom is *true*, whereas the bit value '0' means that the value of the corresponding propositional atom is *false*. For example, at the state 100, we have that the value of the secret of agent $a$ is *true*, while that of both agent $b$ and $c$ are *false*. Notice also the undirected labelled lines used to show the pair of states that are indistinguishable for a given agent. For example, the line labelled 'ac' from state 000 to state 010 shows that both agent $a$ and agent $c$ cannot distinguish between the two states

(this is not surprising as agent $a$ knows the value of its own unique secret, but not that of the unique secret of the other agents; so we see that in both states, agent $a$ knows the same value for $A$, but not for the other atoms; the situation is analogous for agent $c$). Notice also that none of the agents can distinguish between a state and that same state (the accessibility relation for each agent is reflexive).

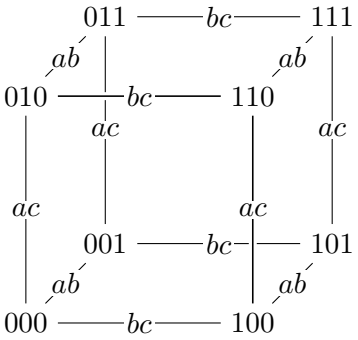



FIGURE 2.1: **Graphical representation of an epistemic model of the initial gossip situation.**

Throughout this thesis we deal with epistemic models or $S5$ models. Given the nature of $S5$ models, there is a set of formulas such that each formula in this set is valid in the class of $S5$ models. We can define a system of logic **S5** based on $S5$ validities, as given in Definition 2.5.

**Definition 2.5.** The logic system **S5**, where we have an operator $K_a$ for every $a \in \mathsf{Ag}$, comprises of all instances of propositional tautologies, the $K$ axiom, the *veridicality* axiom, *positive introspection*, *negative introspection*, and the derivation rules modus ponens and necessitation, as follows:

| | |
|---|---|
| All instances of propositional tautologies | |
| Given $\varphi$ and $\varphi \to \psi$, infer $\psi$ | modus ponens |
| Given $\varphi$, infer $K_a\varphi$ | necessitation |
| $K_a(\varphi \to \psi) \to (K_a\varphi \to K_a\psi)$ | $K$ axiom |
| $K_a\varphi \to \varphi$ | veridicality |
| $K_a\varphi \to K_aK_a\varphi$ | positive introspection |
| $\neg K_a\varphi \to K_a\neg K_a\varphi$ | negative introspection |

The $K$ axiom states that knowledge is closed under consequence. For example if an agent knows that "if today is Sunday then tomorrow is Monday", then, if the agent knows that "today is Sunday" it follows that the agent also knows that "tomorrow is Monday". The veridicality axiom (or *truth* axiom) states that what is known is indeed the case. For example, an agent cannot know that "today is Sunday", if actually it is the case that "today is not Sunday". The positive introspection axiom states that an agent knows *what* it knows, that is, if an agent knows that "an apple is a fruit", then it knows that it knows that "an apple is a fruit". Finally, the negative introspection axiom states that if an agent does not know something, then it also knows that it does not know that thing.

For example, if an agent does not know that “it is raining", then it also knows that it does not know that “it is raining".

### 2.1.2 Group Knowledge

We come now to knowledge among a group of agents. Let $\varphi$ be a formula of the language $\mathcal{L}$. Given a set of agents $\mathsf{Ag}$, we say that a piece of information $\varphi$ is *general knowledge* if and only if every agent in $\mathsf{Ag}$ knows $\varphi$. That is, $E\varphi$, read as “Everybody knows that $\varphi$" holds if and only if $\bigwedge_{a\in\mathsf{Ag}} K_a\varphi$ holds. Of course, the fact that everybody knows a piece of information does not imply that everybody knows that everybody knows that piece of information. That is, $E\varphi$ does not imply $EE\varphi$. Take the example of an email sent to a group of four friends. The message in the email reads “Tomorrow is a holiday”. All four friends read the email privately in their various apartments. But although all four friends know the content of the email, each is still unsure as to whether indeed every one of them knows the content of the email. Each friend considers it possible that one of them has not read the email.

Now let us define iterated general knowledge $E^m$ inductively as follows:

$$\begin{array}{lrcll} \text{[Base Case]} & E^0\varphi & = & \varphi & \\ \text{[Inductive Case]} & E^m\varphi & = & EE^{m-1}\varphi, & \text{for } m \geq 1 \end{array}$$

We can use the definition of iterated general knowledge to illustrate another kind of group knowledge, namely, *common knowledge*, described as the infinite conjunction:

$$C\varphi = \bigwedge_{m=0}^{\infty} E^m\varphi$$

Common knowledge that $\varphi$ (written $C\varphi$) is usually described as “any fool knows that $\varphi$" because it usually holds only for very weak formulas $\varphi$.

But there is yet another notion of group knowledge, namely *distributed knowledge*, which is described as “the wise man knows that $\varphi$”, where it is distributed knowledge that $\varphi$ (written $D\varphi$). This kind of knowledge is implicit in a group, and can be made explicit by aggregating the knowledge of the members of the group. A simple example is set in a group consisting of two agents $a$ and $b$: agent $a$ knows that “if the time is 13:00 then students are at lunch", and agent $b$ knows that “the time is 13:00"; therefore it is distributed knowledge that “students are at lunch".

To define the truth conditions for general knowledge, common knowledge and distributed knowledge on an epistemic model, we begin with the following definition of the relevant language and accessibility relations.

**Definition 2.6** (Language $\mathcal{L}_G$ with group knowledge)**.** Given a set of atomic propositions $\mathsf{P}$, and a set $\mathsf{Ag}$ of agents, the language $\mathcal{L}_G$ is defined by the following BNF grammar:

$$\varphi \quad ::= \quad A \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid K_a\varphi \mid D\varphi \mid E\varphi \mid C\varphi$$

where $A \in \mathsf{P}$ and $a \in \mathsf{Ag}$.

**Definition 2.7.** Given a set $\mathsf{Ag}$ of agents, and an epistemic model $M = (S, \sim, V)$ then:

- Let $\sim_E = \bigcup_{a \in \mathsf{Ag}} \sim_a$
- Let $\sim_D = \bigcap_{a \in \mathsf{Ag}} \sim_a$
- The *reflexive transitive closure* $\sim^*$ of a relation $\sim$ is the smallest relation such that:
  - $\sim \subseteq \sim^*$
  - $s \sim_a^* s$, for every $s \in S$ and for every $a \in \mathsf{Ag}$
  - if $s \sim_a^* t$ and $t \sim_a^* u$ then $s \sim_a^* u$, for all $s, t, u \in S$ and where $a \in \mathsf{Ag}$

**Definition 2.8** (Truth conditions for the language $\mathcal{L}_G$)**.** Given a set of atomic propositions $\mathsf{P}$, and a set $\mathsf{Ag}$ of agents, let $M = (S, \sim, V)$ be an epistemic model. We extend the truth conditions given in Definition 2.4 with the following clauses, to obtain the truth condition for $M, s \models \varphi$, where $\varphi \in \mathcal{L}_G$.

$$\begin{array}{lllll} M, s \models D\varphi & \text{iff} & \text{for every } t, & s \sim_D t \text{ implies } M, t \models \varphi \\ M, s \models E\varphi & \text{iff} & \text{for every } t, & s \sim_E t \text{ implies } M, t \models \varphi \\ M, s \models C\varphi & \text{iff} & \text{for every } t, & s \sim_E^* t \text{ implies } M, t \models \varphi \end{array}$$

**Example 2.2.** *Consider the gossip scenario described in Example 2.1 (a model of that scenario is shown in Figure 2.1). Then we see that everybody knows that agent a knows the value of secret A, that is,* $E(K_a A \vee K_a \neg A)$ *holds in the model for the scenario. It also holds in the model for the scenario that* $EE(K_a A \vee K_a \neg A)$*. Furthermore, it is common knowledge that each agent knows the value of its own unique secret, that is,* $C((K_a A \vee K_a \neg A) \wedge (K_b B \vee K_b \neg B) \wedge (K_c C \vee K_c \neg C))$ *holds in the model for the scenario. It is also common knowledge that each agent knows only its own unique secret. Finally, since each agent knows the value of its own unique secret, we can see that the knowledge of all the secrets in the scenario is distributed knowledge among all the agents in the scenario. So for example, the implication* $(\neg A \wedge B \wedge \neg C) \rightarrow D(\neg A \wedge B \wedge \neg C)$ *also holds in the model for the scenario. Explicit knowledge of all the secrets will be brought about by a sequence of calls among all the agents. The reader is referred to Section 3.3 for more discussion about group notions of knowledge in the gossip scenario.*

## 2.2 Dynamic Epistemic Logic

Dynamic epistemic logic extends the basic epistemic logic to enable the formalisation of knowledge and change of knowledge due to epistemic actions. Hence, in its language, dynamic epistemic logic includes a new type of modality called action modality. The action modality gives rise to dynamic constructs in the language, and such dynamic

constructs are used to express an epistemic action together with an epistemic property that is brought about due to the execution of the epistemic action.

In its semantics, dynamic epistemic logic gives meaning to the dynamic constructs in the language. In both the language and the semantics of dynamic epistemic logic that we will present, we focus on an approach that expresses an action using an *action model*. Hence we refer to this logic as action model logic.

We begin now by presenting the definition of action models, and in the subsections that follow, we present the syntax and semantics of action model logic.

**Definition 2.9** (Action Model)**.** Given a set $\mathsf{Ag}$ of agents and a set $\mathsf{P}$ of atomic propositions, an action model is a structure $\mathsf{M} = (\mathsf{S}, \sim, \mathsf{pre})$, where $\mathsf{S}$ is a domain of *action points*, and $\sim_{\mathsf{a}}$ is an equivalence relation on $\mathsf{S}$ for each $a \in \mathsf{Ag}$, and $\mathsf{pre} : \mathsf{S} \to \mathcal{L}$ is a function that takes an action point $\mathsf{s} \in \mathsf{S}$ and returns a precondition $\mathsf{pre}(\mathsf{s}) \in \mathcal{L}$. A *pointed action model* is a structure $(\mathsf{M}, \mathsf{s})$ where $\mathsf{s} \in \mathsf{S}$.

### 2.2.1 Syntax of Action Model Logic

**Definition 2.10** (Language of action model logic)**.** Given a set $\mathsf{Ag}$ of agents, and a set of propositional atoms $\mathsf{P}$, the language $\mathcal{L}_{AM}$ of action model logic is defined as follows:

$$\begin{array}{rlll} \mathcal{L}_{AM} \ni & \varphi & ::= & A \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \to \varphi) \mid K_a\varphi \mid D\varphi \mid E\varphi \mid C\varphi \mid [\alpha]\varphi \\ & \alpha & ::= & (\mathsf{M}, \mathsf{s}) \mid (\alpha \cup \alpha) \end{array}$$

where $a \in \mathsf{Ag}$, and $A \in \mathsf{P}$, and $(\mathsf{M}, \mathsf{s})$ is a pointed action model, with $\mathsf{M}$ having a finite domain of action points, and where $\mathsf{pre}(\mathsf{s}) \in \mathcal{L}$.

### 2.2.2 Semantics of Action Model Logic

**Definition 2.11** (Semantics of the formulas of $\mathcal{L}_{AM}$)**.** Let $(M, s)$ be a pointed epistemic model, where $M = (S, \sim, V)$, and let $\mathsf{M} = (\mathsf{S}, \sim, \mathsf{pre})$ be an action model. Then the truth conditions of $\varphi, \psi \in \mathcal{L}_{AM}$ is defined inductively as follows:

$$\begin{array}{lll} M, s \models A & \text{iff} & V_s(A) = 1 \\ M, s \models \neg\varphi & \text{iff} & M, s \not\models \varphi \\ M, s \models \varphi \wedge \psi & \text{iff} & M, s \models \varphi \text{ and } M, s \models \psi \\ M, s \models \varphi \vee \psi & \text{iff} & M, s \models \varphi \text{ or } M, s \models \psi \\ M, s \models \varphi \to \psi & \text{iff} & M, s \models \neg\varphi \text{ or } M, s \models \psi \\ M, s \models K_a\varphi & \text{iff} & \text{for all } t,\ s \sim_a t \text{ implies } M, t \models \varphi \\ M, s \models D\varphi & \text{iff} & \text{for all } t,\ s \sim_D t \text{ implies } M, t \models \varphi \\ M, s \models E\varphi & \text{iff} & \text{for all } t,\ s \sim_E t \text{ implies } M, t \models \varphi \\ M, s \models C\varphi & \text{iff} & \text{for all } t,\ s \sim_E^* t \text{ implies } M, t \models \varphi \\ M, s \models [\alpha]\,\varphi & \text{iff} & \text{for all } (M^{\mathsf{M}}, (s, \mathsf{s})),\ (M, s)[\![\alpha]\!](M^{\mathsf{M}}, (s, \mathsf{s})) \text{ implies } M^{\mathsf{M}}, (s, \mathsf{s}) \models \varphi \end{array}$$

where $\sim_D$, $\sim_E$ and $\sim_E^*$ are as defined in Definition 2.7; and where $[\![\alpha]\!]$ is a binary relation such that:

$$
\begin{array}{ll}
[\![\alpha \cup \alpha']\!] = [\![\alpha]\!] \cup [\![\alpha']\!], \text{ and} & \\
(M,s)[\![\mathsf{M},\mathsf{s}]\!](M^{\mathsf{M}},(s,\mathsf{s})) \quad \text{iff} & M^{\mathsf{M}} = (S^{\mathsf{M}}, \sim^{\mathsf{M}}, V^{\mathsf{M}}) \text{ is given by:} \\
& S^{\mathsf{M}} = \{(s,\mathsf{s}) \mid s \in S,\ \ \mathsf{s} \in \mathsf{S}, \text{ and } M, s \models \mathsf{pre}(\mathsf{s})\} \\
& (s,\mathsf{s}) \sim_a^{\mathsf{M}} (t,\mathsf{t}) \text{ iff } s \sim_a t \text{ and } \mathsf{s} \sim_a \mathsf{t} \\
& V^{\mathsf{M}}_{(s,\mathsf{s})}(A) = V_s(A)
\end{array}
$$

The semantics of action model logic is used to derive a model of the gossip situation resulting from executing an action at an initial gossip situation. In the semantics of action model logic, we represent factual states in a model of the resulting gossip situation as pairs consisting of (a) name of the factual state in the model for the initial gossip situation, and (b) name of the action point executed in that factual state in the model for the initial gossip situation. For example, for the factual state $(s,\mathsf{s})$, the letter $s$ is the name of the factual state in the initial model, and the letter $\mathsf{s}$ is the name of the action point executed in the factual state $s$. The only $(s,\mathsf{s})$ pairs that are allowed in the domain $S^{\mathsf{M}}$ of the resulting model $M^{\mathsf{M}}$ are pairs such that the action $\mathsf{s}$ can be executed in $s$. That is, $M, s \models \mathsf{pre}(\mathsf{s})$. Considering the points (or the factual states) of the models, the valuation $V^{\mathsf{M}}_{(s,\mathsf{s})}$ of the factual state $(s,\mathsf{s})$ in the resulting model is equal to the valuation of the state $s$ in the initial model (we consider actions that do not change the facts of the real world, but can only change the epistemic property of agents). Furthermore, the names given to such factual states are not relevant. What matters is which facts are true in those states and how those states relate to other states.

Regarding the accessibility relations, two new states are indistinguishable for an agent if and only if they result from two indistinguishable action points executed at two indistinguishable states.

Consider again the initial gossip situation described in Example 2.1, the epistemic model of which is illustrated in Figure 2.1 (the model is also shown in the left hand side of the transition in Figure 2.2). Recall that we omit reflexive and transitive accessibility links in the drawing of epistemic models, for the sake of visual clarity. Let agent $a$ and agent $b$ call each other in this initial gossip situation. (Assume that it is common knowledge among all the agents that they are in the initial gossip situation (see Example 2.1 and 2.2); also assume that the agents have common knowledge about whom is calling who, although the agent who is not involved in a call does not know exactly the value of the secrets exchanged in the call). The resulting gossip situation due to this call $ab$ is shown at the right hand side of the transition in Figure 2.2. We model the call $ab$ as an action model, and use the semantics of action model logic to derive the model of the resulting gossip situation due to the $ab$ call at the initial gossip situation. Note that at the initial gossip situation, both agent $a$ and agent $b$ are uncertain about the value of each other's secret, and they are both uncertain about the value of secret $C$. However, the agents know that during the $ab$ call at the initial gossip situation, both agent $a$ and

agent $b$ can only learn the value of each other's secret, but both will remain ignorant of the value of secret $C$ after the $ab$ call.

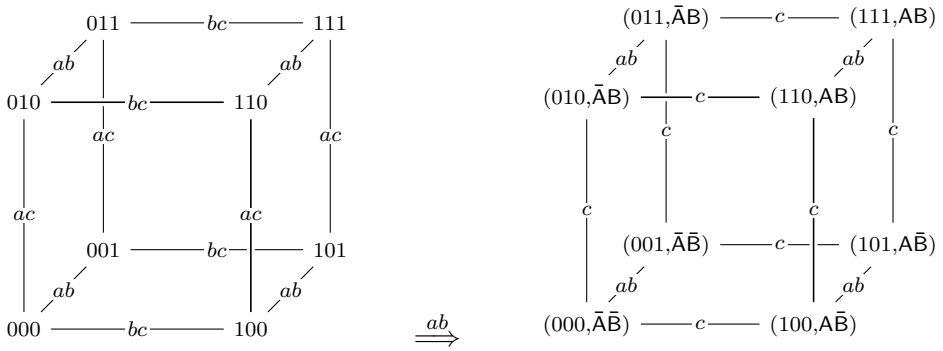



FIGURE 2.2: **State transition after an $ab$ call at the initial gossip situation.**

From agent $c$'s point of view, during the $ab$ call, agent $a$ could be learning that $B$ is true, or it could be learning that $B$ is false. Also, agent $b$ could be learning that $A$ is true, or it could be learning that $A$ is false. But agent $c$ cannot distinguish between the four actions, namely: (i) agent $b$ learns $\neg A$ and agent $a$ learns $\neg B$ (action point $\bar{A}\bar{B}$), (ii) agent $b$ learns $A$ and agent $a$ learns $\neg B$ (action point $A\bar{B}$), (iii) agent $b$ learns $\neg A$ and agent $a$ learns $B$ (action point $\bar{A}B$), and (iv) agent $b$ learns $A$ and agent $a$ learns $B$ (action point $AB$).

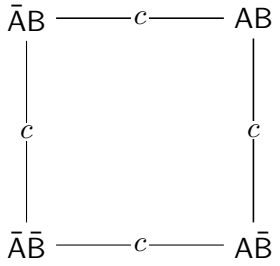



FIGURE 2.3: **Action model for $ab$ call at the initial gossip situation.**

From agent $a$'s point of view, either agent $a$ learns $\neg B$ or agent $a$ learns $B$. And from agent $b$'s point of view, either agent $b$ learns $\neg A$ or it learns $A$. Moreover, agent $b$ knows the value of secret $B$ and agent $a$ knows the value of secret $A$. Also, both agent $a$ and agent $b$ can distinguish between any pair among the four action points $\bar{A}\bar{B}$, $\bar{A}B$, $A\bar{B}$ and $AB$, since they are both involved in each of the actions, and know which is the actual action point. The preconditions for the four action points are as follows:

$$\begin{aligned}
\mathsf{pre}(\bar{A}\bar{B}) &= K_a\neg A \wedge K_b\neg B \\
\mathsf{pre}(\bar{A}B) &= K_a\neg A \wedge K_bB \\
\mathsf{pre}(A\bar{B}) &= K_aA \wedge K_b\neg B \\
\mathsf{pre}(AB) &= K_aA \wedge K_bB
\end{aligned}$$

The action model for the $ab$ call at the initial gossip situation is shown in Figure 2.3. Note that each agent cannot distinguish an action point and that same action point. So

the accessibility relation for each agent in the action model is reflexive. But we have chosen to omit these reflexive accessibility links (and the transitive accessibility links) in the drawing of action models, again, for visual clarity.

To illustrate the accessibility relations, consider that in the initial gossip situation agent $c$ cannot distinguish between a factual state where secret $A$ is true and another factual state where secret $A$ is false, if secret $C$ is the same in both states. Also, in the action model for the $ab$ call, agent $c$ cannot distinguish between an action point wherein agent $b$ is learning that $A$ is true, and another action point wherein agent $b$ is learning that $A$ is false. Since agent $c$ cannot distinguish between the said action points, and since it cannot distinguish between the said factual states, if the said action points are each executed in the said factual states, then agent $c$ can also not distinguish between any two of the resulting factual states. Particularly, the factual state $(000, \bar{\mathsf{A}}\bar{\mathsf{B}})$ in the model for the resulting gossip situation (right hand side of the transition in Figure 2.2) is the result of executing the action $\bar{\mathsf{A}}\bar{\mathsf{B}}$ in the factual state 000 in the model for the initial gossip situation (left hand side of the transition in Figure 2.2). Also, the factual state $(100, \mathsf{A}\bar{\mathsf{B}})$ in the model for the resulting gossip situation is the result of executing action $\mathsf{A}\bar{\mathsf{B}}$ at the state 100 in the model for the initial gossip situation. Notice that the two action points $\bar{\mathsf{A}}\bar{\mathsf{B}}$ and $\mathsf{A}\bar{\mathsf{B}}$ are indistinguishable for agent $c$, and the two factual states 000 and 100 in the model for the initial gossip situation are indistinguishable for agent $c$. Notice also how the preconditions of the said action points are satisfied in the factual states where they are executed, and see that indeed, agent $c$ cannot distinguish between state $(000, \bar{\mathsf{A}}\bar{\mathsf{B}})$ and state $(100, \mathsf{A}\bar{\mathsf{B}})$ in the model for the resulting gossip situation.

On the other hand, even though agent $a$ could not initially distinguish between state 000 and state 010 in the model for the initial gossip situation, we see that agent $a$ can distinguish between the action point wherein agent $a$ learns that $B$ is false and agent $b$ learns that $A$ is false (that is, action point $\bar{\mathsf{A}}\bar{\mathsf{B}}$), and the action point wherein agent $a$ learns that $B$ is true and $b$ learns that $A$ is false (that is, action point $\bar{\mathsf{A}}\mathsf{B}$). Therefore, agent $a$ can distinguish between the factual states resulting from the execution of both of those action points in the initial gossip situation, namely state $(000, \bar{\mathsf{A}}\bar{\mathsf{B}})$ and state $(010, \bar{\mathsf{A}}\mathsf{B})$ in the model for the resulting gossip situation. Finally, using the semantics of the knowledge operator given in Definition 2.11, it is easy to verify at any factual state in the model for the resulting gossip situation, that both agent $a$ and agent $b$ know the value of each other's secret, and that they both remain ignorant of the value of agent $c$'s secret, whereas agent $c$ remains ignorant of the value of the secret of both agent $a$ and agent $b$.

However, agent $c$ also learns something new after the $ab$ call. For example, after the $ab$ call agent $c$ learns that agent $a$ now knows the value of $b$'s secret. That is, $K_c(K_aB \vee K_a\neg B)$ holds in all the states in the model of the resulting gossip situation, and for example, we demonstrate this for one of the states in the resulting model. Consider state $(000, \bar{\mathsf{A}}\bar{\mathsf{B}})$ in the model for the resulting gossip situation. To check that $K_c(K_aB \vee K_a\neg B)$ holds in $(000, \bar{\mathsf{A}}\bar{\mathsf{B}})$ we follow the semantics given in Definition 2.4. At $(000, \bar{\mathsf{A}}\bar{\mathsf{B}})$, the

factual states that are indistinguishable for agent $c$ are as follows: $(000, \bar{\mathsf{A}}\bar{\mathsf{B}})$, $(010, \bar{\mathsf{A}}\mathsf{B})$, $(100, \mathsf{A}\bar{\mathsf{B}})$ and $(110, \mathsf{AB})$ (these factual states respectively resulted from the factual states 000, 010, 100 and 110 in the model for the initial gossip situation). We now show that it is not the case that $(K_aB \vee K_a\neg B)$ holds in all of the states 000, 010, 100 and 110, but on the other hand $(K_aB \vee K_a\neg B)$ holds in all of the states $(000, \bar{\mathsf{A}}\bar{\mathsf{B}})$, $(010, \bar{\mathsf{A}}\mathsf{B})$, $(100, \mathsf{A}\bar{\mathsf{B}})$ and $(110, \mathsf{AB})$. That is, we show that at state 000 agent $c$ does not *know* that agent $a$ *knows* the value of $b$'s secret, but after the $ab$ call, at state $(000, \bar{\mathsf{A}}\bar{\mathsf{B}})$, agent $c$ *knows* that agent $a$ *knows* the value of $b$'s secret. We now proceed as follows.

At 000, agent $a$ cannot distinguish between 000 and 010, and these two states differ in their respective values for secret $B$. That is, at 000, agent $a$ considers it possible that the actual state is 010 (that is, agent $a$ considers it possible that the value of $b$'s secret is *true*), and also agent $a$ considers it possible that the actual state is 000 (that is, agent $a$ also considers it possible that the value of $b$'s secret is *false*), so agent $a$ does not *know* the value of secret $B$ at 000. In turn, since 000 is accessible to 000 for agent $c$, we also see that $\neg K_c(K_aB \vee K_a\neg B)$ holds at 000. But at $(000, \bar{\mathsf{A}}\bar{\mathsf{B}})$, agent $a$ can distinguish between $(000, \bar{\mathsf{A}}\bar{\mathsf{B}})$ and $(010, \bar{\mathsf{A}}\mathsf{B})$. So, for agent $a$, the only possible value for secret $B$ at $(000, \bar{\mathsf{A}}\bar{\mathsf{B}})$ is *false*, and therefore agent $a$ *knows* $\neg B$ at $(000, \bar{\mathsf{A}}\bar{\mathsf{B}})$. On the other hand, the states that agent $a$ cannot distinguish from 010, namely, 010 and 011, have the same value (namely, *true*) for secret $B$, and this indistinguishability is maintained between states $(010, \bar{\mathsf{A}}\mathsf{B})$ and $(011, \bar{\mathsf{A}}\mathsf{B})$, that is, in the factual states that resulted from 010 and 011 after the $ab$ call. Likewise the states that agent $a$ cannot distinguish from 100, namely, 100 and 101, have the same value (namely, *false*) for secret $B$, and this indistinguishability is also maintained between states $(100, \mathsf{A}\bar{\mathsf{B}})$ and $(101, \mathsf{A}\bar{\mathsf{B}})$. Finally, the states that agent $a$ cannot distinguish from 110, namely, 110 and 111, have the same value (namely, *true*) for secret $B$, and this indistinguishability is also maintained between states $(110, \mathsf{AB})$ and $(111, \mathsf{AB})$.

From the foregoing, we see that in $(000, \bar{\mathsf{A}}\bar{\mathsf{B}})$ indeed agent $c$ *knows* that agent $a$ cannot distinguish only those pair of factual states where the value of secret $B$ is the same. And this is equivalent to saying that $K_c(K_aB \vee K_a\neg B)$ holds in $(000, \bar{\mathsf{A}}\bar{\mathsf{B}})$, following the semantics given in Definition 2.4.

We conclude here our presentation of some of the basics of epistemic and dynamic epistemic logic, we now turn briefly to another topic in order to present other background definitions for later chapters of the thesis.

## 2.3 Network Topologies

In Chapters 5 and Chapter 6, particularly, we study epistemic gossip protocols in the setting of various network topologies. The following are some of the basic concepts from graph theory which we employ.

**Definition 2.12** (Graph)**.** A *graph* is a tuple $G = (V, E)$, where $V$ is a finite set of nodes (or vertices), and $E$ is a finite set of edges such that $E \subseteq V \times V$.

We assume that the edges of $G$ are *undirected*. When describing an edge of a graph we also write $uv$ for the pair $(u, v)$, where $u, v \in V$ and $(u, v) \in E$. If $uv \in E$ then $u$ and $v$ are said to be *adjacent vertices*. Such $u$ is a *neighbour* of $v$ (and $v$ a neighbour of $u$ since the edges are undirected). We assume $G$ is a *simple graph* in which case no edge joins any node to itself and there is only one edge between any two nodes in $V$, where both $V$ and $E$ are non-empty, and $G$ is *non-trivial*, that is, the size of the set $V$ is greater than one. There is a *path* between two nodes $u_0, u_m \in V$ if and only if there is some sequence $\langle u_0u_1, u_1u_2, \ldots, u_{m-1}u_m\rangle$ consisting of edges in $E$. Finally, we assume $G$ is *connected*, that is, for every pair of nodes $u, v \in V$, there is a path between $u$ and $v$ [62].

Consider a gossip scenario in which the set of agents is $\mathsf{Ag}$, where $|\mathsf{Ag}| = n$. Let us describe the network of such agents using the graph $G = (V, E)$, as follows.

**Definition 2.13** (Network Topology)**.** Let $V = \mathsf{Ag}$, and let an edge $a_ia_j \in E$ between any pair of agents denote that there is a communication link between agent $a_i$ and $a_j$. Then we define various topologies of such networks as the type of graph $G$, as follows:

- **Circle Topology Network**. Let $E = \{(a_1, a_2),\ (a_2, a_3),\ \ldots,\ (a_k, a_{k+1}),\ \ldots,\ (a_n, a_1)\}$, (for all $3 \leq k < n$), and let $E^s$ be the symmetric closure of $E$. We define the graph $G = (V, E^s)$ as a circle topology network. For an illustration, see Figure 2.4(a).

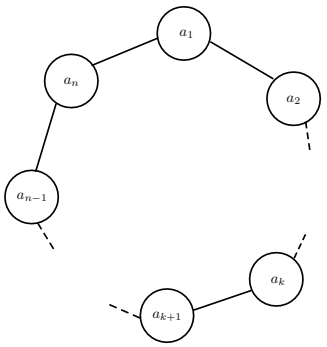
- **Line Topology Network**. Let $E = \{(a_1, a_2),\ (a_2, a_3),\ \ldots,\ (a_k, a_{k+1}),\ \ldots,\ (a_{n-1}, a_n)\}$, (for all $3 \leq k < n-1$), and let $E^s$ be the symmetric closure of $E$. We define the graph $G = (V, E^s)$ as a line topology network. For an illustration, see Figure 2.4(e).

- **Star Topology Network**. Let $E = \{(a_1, a_2), (a_1, a_3), \ldots, (a_1, a_k), \ldots, (a_1, a_n)\}$, (for all $3 < k < n$), and let $E^s$ be the symmetric closure of $E$. We define the graph $G = (V, E^s)$ as a star topology network. For an illustration, see Figure 2.4(c).

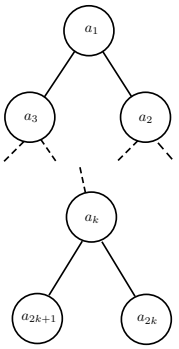
- **Binary Tree Topology Network**. Let $E = \{(a_1, a_2),\ (a_1, a_3),\ \ldots,\ (a_k, a_{2k}),\ (a_k, a_{2k+1}),\ \ldots\}$, (for all $1 < k \leq \lceil \frac{n-1}{2} \rceil$), and let $E^s$ be the symmetric closure of $E$. We define the graph $G = (V, E^s)$ as a binary tree topology network. For an illustration, see Figure 2.4(b).

- **Complete Topology Network**. A complete topology network is obtained when for every pair $a_j, a_k \in V$ there is a corresponding edge $a_ja_k$ in $E$ of the graph $G$. For an illustration, see Figure 2.4(d).

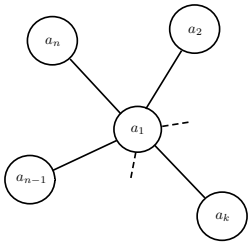
Finally, when the graph $G$ consists of the vertices and edges of a tree of $n$ nodes, we will generally refer to such network as a *Tree Topology Network*. On the other hand, a graph $G$ is a *Connected Topology Network* if and only if $G$ is connected. And, a network topology that is not a complete topology network is also said to be an *Incomplete Topology Network*.



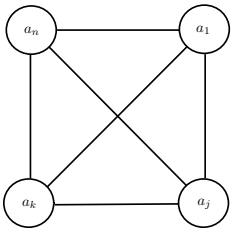

(a) Circle Topology



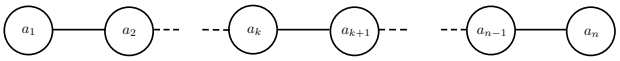

(b) Binary Tree Topology




(c) Star Topology




(d) Complete Topology




(e) Line Topology

FIGURE 2.4: **Network topologies.**

## 2.4 Summary of Chapter

In this chapter we provided some logical background and presented some notation that will be used throughout this thesis. We Introduced a basic epistemic logic for describing scenarios in which there is no change in the knowledge of agents. We then went further and showed one way that this basic logic can be extended to account for change of knowledge of agents due to epistemic actions. In Chapter 3 especially, we build on these ideas to create a logical formalism for describing and reasoning about epistemic gossip protocols. Finally, from graph theory we presented some definitions, notation and assumptions that we rely on in the material presented in the later chapters of this thesis.

# Chapter 3

# Epistemic Protocols for Gossip

## 3.1 Introduction

In Chapter 1 we introduced the gossip problem and described its solution for a scenario comprising of $n \geq 1$ agents. Particularly we described the *Fixed Schedule*, which distributes all the secrets in a gossip scenario among all the agents in $2n - 4$ rounds. For ease of reference let us repeat the Fixed Schedule here.

Consider a gossip scenario with agents $a, b, c, d, e, f \ldots$. Then the Fixed Schedule is as follows.

**Fixed Schedule** Choose four agents from the set of agents $\mathsf{Ag}$, say $a, b, c, d$, and one of those four, say $a$. First, agent $a$ makes one call to each of the agents in $\mathsf{Ag} \setminus \{a, b, c, d\}$. Then, the Four-agent protocol is executed among agents $a, b, c, d$ (say, the calls $ab; cd; ac; bd$ are made). Finally, agent $a$ again makes one call to each of the agents in $\mathsf{Ag} \setminus \{a, b, c, d\}$.

An example execution sequence of the Fixed Schedule is as follows:

$$ae; af; ab; cd; ac; bd; ae; af$$

In the Fixed Schedule, the order of the calls is pre-designated. In this chapter we describe epistemic (or knowledge-based) protocols for gossip, wherein an agent calls another agent, not because it has been instructed to do so, but based on its knowledge. We begin, in Section 3.2 and in Section 3.3, with an informal discussion of these protocols. In Sections 3.4 and 3.5, we present logical formalisms for describing and reasoning about epistemic gossip protocols, and pairwise calls in a gossip scenario. And in Section 3.6 we present a formalisation of epistemic gossip protocols introduced in Section 3.2, and present some logical properties of these protocols. We conclude the chapter in Section 3.7.

## 3.2 Epistemic Gossip Protocols

A gossip protocol can be described as a rule that expresses the condition under which an agent can communicate with another agent by means of a *telephone call* within the gossip scenario. Following this way of thinking, we can describe an epistemic gossip protocol in terms of a rule that expresses some *knowledge-based calling condition* required for an agent to call another agent in the scenario. Such a calling condition is based on what an agent knows about an agent, or about an agent's knowledge of the secrets in the scenario. The idea is that in each round of gossiping, each agent makes use of the calling condition given by its protocol to determine whom it can call in the round. When the calling condition of a protocol enables an agent to call more than one agent in a round, the agent randomly chooses one *callee* among those agents whom it is enabled to call. Since each agent executes its own protocol, more than one agent may be enabled to make a call in each round of gossiping, in that case we assume an arbiter who randomly chooses one agent among all the agents who are enabled to make a call in any given round. The goal is that after a series of rounds, every agent knows the secret of every other agent in the scenario. In the rest of this section we informally describe and discuss a number of epistemic gossip protocols, and we begin with a protocol which we call the Learn New Secrets protocol, as follows.

**Learn New Secrets** For any agent $a_i$, the protocol for $a_i$ is: choose an agent $a_j$ such that $a_i$ does not know the secret of $a_j$, and let $a_i$ call $a_j$.

It is easy to see that this protocol will achieve the goal that everybody knows every secret. No call sequence obtained from the Fixed Schedule can be obtained by the Learn New Secrets protocol. The agent who initiates the call in each of the final two calls of the Fixed Schedule, calls another agent whose secret it already knows. For example, in the execution sequence of the Fixed Schedule which we considered earlier, in the final two calls $(ae; af)$ agent $a$ calls agents whose secret it already knows. But the same information transitions* can be achieved by an execution sequence of Learn New Secrets: instead of final calls $ae; ef$, make final calls from agent $e$ and agent $f$ to agent $b$ (or to agent $c$, or to agent $d$): $eb; fb$. This is allowed by the protocol, as agent $e$ and agent $f$ do not know the secret of agent $b$ at the time of that call. The Learn New Secrets protocol also allows for longer execution sequences than the Fixed Schedule. For example, for $n = 4$, the following is an execution sequence of Learn New Secrets: $ab; ac; ad; bc; bd; cd$. The longest execution sequence of Learn New Secrets is given by $n(n-1)/2$ (we prove this later, in Chapter 6), and the given execution sequence is an example of such a longest execution sequence.

*See Table 1.1 for an example of information transitions due to a call sequence.

### Possible and Known Information Growth protocols

In the execution sequence of the Fixed Schedule given earlier, agent $a$ called the same agents at the end as at the beginning. When agent $a$ first called agent $e$, agent $a$ learnt secret $E$ and agent $e$ learnt secret $A$. When agent $a$ called agent $e$ again, agent $e$ learnt secrets $B, C, D$, and $F$ from agent $a$. One can say that agent $a$ called agent $e$ again because agent $a$ learnt some new secret in the intervening calls which it knows agent $e$ have not learnt. In other words, agent $a$ *knows* that agent $e$ will learn some new secret if agent $a$ calls agent $e$ again. This feature can also be used in an epistemic protocol. Let us define '$A_k$ is learnt in the call $a_i a_j$' as 'before the call, only one of $a_i$ and $a_j$ did *not* know $A_k$, but after the call both $a_i$ and $a_j$ know $A_k$'. Now consider these protocols.

**Known Information Growth de Dicto** For any agent $a_i$, the protocol for $a_i$ is: choose an agent $a_j$ such that $a_i$ knows that there is some secret $A_k$ that would be learnt in the call $a_i a_j$, and let $a_i$ call $a_j$.

**Possible Information Growth de Dicto** For any agent $a_i$, the protocol for $a_i$ is: choose an agent $a_j$ such that $a_i$ considers it possible that there is some secret $A_k$ that would be learnt in the call $a_i a_j$, and let $a_i$ call $a_j$.

The Possible Information Growth de Dicto protocol may loop and therefore termination is not guaranteed. For example, for four agents, the following is an infinite execution sequence: $ab; cd; ab; cd; ab; \ldots$ (that is, the sub-sequence $ab; cd$ repeats infinitely). In the third round, $a$ considers it possible that $b$ has learnt something new, namely if the second call had been one of $bc, bd, cb, db$. Therefore, after $ab; cd$, call $ab$ can be chosen according to the protocol. We could also say that in the third round agent $a$ is unable to distinguish the call sequences $ab; cd$, $ab; dc$, $ab; bc$, $ab; cb$, $ab; bd$, $ab; db$.

At first sight, the Possible Information Growth de Dicto protocol seems to have an advantage over the Known Information Growth de Dicto protocol. Maybe there are situations wherein after a certain number of calls, due to the uncertainty about who called who, no agent knows for certain that calling any other agent will result in information growth. That would cause a deadlock in the Known Information Growth de Dicto protocol, from which an agent can still escape when using the Possible Information Growth de Dicto protocol. But on second sight, such a situation cannot occur. Consider any situation wherein it is not yet the case that all agents know all secrets. Then there is an agent $a_i$ who does not know secret $A_j$. Agent $a_i$ *knows* that when it calls agent $a_j$, it will learn $A_j$. So the knowledge condition that $a_i$ knows that there is a secret $A_k$ that is learnt in the call $a_i a_j$, is fulfilled, namely for $A_k = A_j$.

Now, consider the following 'de Re' variation of the Known Information Growth protocol. A similar variation exists for the Possible Information Growth protocol. 'De Re' and 'de Dicto' knowledge are considered in e.g., [1, 34, 64].

**Known Information Growth de Re** For any agent $a_i$, the protocol for $a_i$ is: choose an agent $a_j$ such that $a_i$ knows of a secret $A_k$ that would be learnt in the call $a_i a_j$, and let $a_i$ call $a_j$.

**Possible Information Growth de Re** For any agent $a_i$, the protocol for $a_i$ is: choose an agent $a_j$ such that there is a secret $A_k$ of which $a_i$ considers possible that it will be learnt in the call $a_i a_j$, and let $a_i$ call $a_j$.

Consider the call sequence: $ab; cd; ac; ab$, where the set of agents in the scenario is $\{a, b, c, d\}$. Let us assume that the system is synchronous, that is, calls are made at fixed intervals. Clearly, the given call sequence is not a possible execution sequence of the Learn New Secrets protocol, because two agents will never call each other twice in an execution sequence of the Learn New Secrets protocol. But the given call sequence *is* a possible execution sequence of the Known Information Growth de Re protocol. Consider the point of view of agent $b$. After the initial call $ab$, agent $b$ is not involved in the two subsequent calls (since the system is synchronous, it *knows* that it missed two calls). The second and third calls must therefore have been among $a$, $d$ and $c$. At the fourth round, the following histories are then possible from the perspective of agent $b$—for now, in order to keep the following list of possible histories brief, let us identify calls $ab$ with $ba$:

$$\begin{array}{ll} ab; ac; ad & (i) \\ ab; ac; cd & (ii) \\ ab; ad; ac & (iii) \\ ab; ad; cd & (iv) \\ ab; cd; ac & (v) \\ ab; cd; ad & (vi) \end{array}$$

Now, the fourth round call is between agent $a$ and agent $b$. So following each of the histories in cases (i) through (vi), agent $b$ will learn new secrets from agent $a$ in the fourth round call: namely $C, D$ in (i), $C$ in (ii), $C, D$ in (iii), $D$ in (iv), $C, D$ in (v), and $C, D$ in (vi). Therefore, agent $b$ knows that there is a secret (namely $C$ or $D$) that it will learn by calling agent $a$ again. Thus, the epistemic calling condition of Known Information Growth de Dicto is satisfied for agent $b$ to call agent $a$ in the fourth round. However, agent $b$ does not know that it will learn secret $C$ by calling agent $a$, and agent $b$ also does not know that it will learn secret $D$ by calling agent $a$. So, it is not the case that there is a secret such that agent $b$ knows that it will learn that secret by calling agent $a$. Therefore, the epistemic calling condition of Known Information Growth de Re is not satisfied for agent $b$ to call agent $a$ in the fourth round.

Although we chose, for the sake of brevity, to identify calls $ab$ and $ba$ in the foregoing example of possible histories for agent $b$, note however that in reasoning about possible histories of a protocol it is important to distinguish between such symmetric calls. The reason is that there are situations where the call $ab$, for example, may not be possible, that is, the calling condition for agent $a$ to initiate a call to agent $b$ does not hold,

although the call $ba$ is possible. To illustrate this point, consider the following example execution sequence of Learn New Secrets: $bc; ac$. In the third round, agent $a$ can no longer call agent $b$ since agent $a$ already knows the secret of agent $b$. But agent $b$ can call agent $a$ in the third round since at that point agent $b$ does not yet know the secret of agent $a$.

Many more variations of epistemic gossip protocols are possible, for example, (i) epistemic gossip protocols with uncertainty over the number of calls that have taken place (asynchronous systems), such that only after the fourth call in the sequence $ab; cd; ac; ab$ does agent $b$ learn that two intervening calls must have taken place after its first call with agent $a$; and (ii) epistemic gossip protocols for rounds of parallel calls, wherein, for five agents $a, b, c, d, e$, agent $e$ learns that two calls must be taking place in a given round when it finds every other agent engaged (so it is unable to distinguish rounds $\{ab, cd\}$, $\{ac, bd\}$, $\{ad, bc\}$). We do not explore protocols with parallel calls in this thesis, nor do we investigate asynchronous systems.

## 3.3 Knowledge and Gossip

What sort of knowledge do the agents obtain in these gossip protocols? This becomes interesting if we do not only consider what agents know about the secrets but also consider what they know about each other. In this section, we give an informal overview of: knowledge in the initial state of information (wherein every agent only knows its own secret), the change of knowledge due to a call between two agents, and the knowledge conditions after termination of a protocol.

**Initial State of Information** We can represent the uncertainty of the agents about their secrets in a kind of multi-agent Kripke model, namely, an epistemic model[†] (see Section 2.1.1 for some background on Kripke models). Let us consider agent $a$'s secret as a proposition $A$ of which the value is initially only known by agent $a$. So we depict a model for a scenario consisting of three agents $a, b, c$ and at the initial state of information (or initial gossip situation) as shown in Figure 3.1, where a node like 011 stands for '$A$ is false and $B$ is true and $C$ is true' (in the order $\langle a, b, c\rangle$ the digits 0 and 1 stand for the value of the propositions $A, B, C$, respectively).

This is a rather standard kind of situation in epistemic logic. The secret of an agent is its local state, and every agent only knows its local state (and this is common knowledge). For example, in state 011 we have that agent $a$ knows that $A$ is false (because $A$ is false in 011, 010, 001 and 000, the four states considered possible by $a$), and that agent $b$ knows that $B$ is true and that agent $c$ knows that $C$ is true. We can also represent the distribution of secrets over agents as a list (or, as a function from agents to subsets of the set of all secrets). The situation in Figure 3.1 is succinctly represented by $A.B.C$, whereas the situation in Figure 3.2 is represented as $AB.AB.C$.

[†]Note that in the graphical depiction of $S5$ models throughout this thesis we omit the reflexive and transitive accessibility links for the sake of visual clarity.

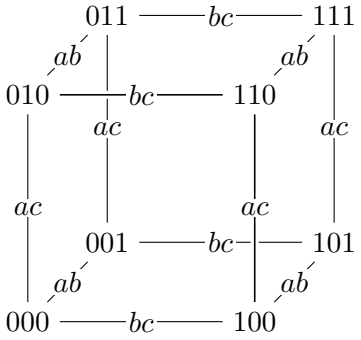



FIGURE 3.1: **The uncertainty of agents about the secrets at the initial gossip situation.**

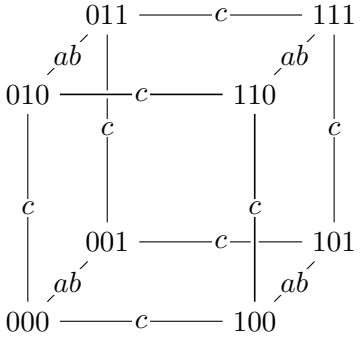



FIGURE 3.2: **Epistemic model for the gossip situation AB.AB.C.**

**Executing a Phone Call** Let us now execute a telephone call in this setting. We get from $A.B.C$ to $AB.AB.C$ by executing the call $ab$. What sort of dynamics is a telephone call? A telephone call is a very different form of communication than an announcement in the presence of other agents. An announcement is *public*. This means that, after agent $a$ says 'The old name of Chennai is Madras' in the presence of agent $b$ and agent $c$, then agent $b$ knows that the old name of Chennai is Madras, but agent $c$ also knows that agent $b$ knows that, and agent $a$ knows that agent $c$ knows that agent $b$ knows that, and so on. The information that the old name of Chennai is Madras, is common knowledge between the three agents. But if first agent $a$ calls agent $b$ to tell him that, and then agent $b$ calls agent $c$, all three know that the old name of Chennai is Madras, but it is not common knowledge. It is even impossible that this becomes common knowledge if nothing is known about the timing of the phone calls [67, Section 2.3].

Instead of the mere transition for the call $ab$ we list those for the sequence $ab; ac; bc$. The corresponding transitions between the gossip situations, that is, the list of who knows what secrets are as follows (the same transitions are depicted in Figure 3.3):

$$A.B.C \overset{ab}{\to} AB.AB.C \overset{ac}{\to} ABC.AB.ABC \overset{bc}{\to} ABC.ABC.ABC$$

Now here is an obvious but surprising observation. Having first explained that calls do not create common knowledge of the secrets, after all, at the end of the execution sequence $ab; ac; bc$, there is common knowledge that all three agents know all the secrets.

We can understand this as follows: the agents have common knowledge what protocol is being carried out (the agents also have common knowledge about the number of agents in the scenario). In this case, the call sequence could be an execution sequence of the Fixed Schedule, but also an execution sequence of the Learn New Secrets protocol. On the assumption of synchronisation, if there are three agents and the second call is $ac$, then agent $b$ knows that the call $ac$—or $ca$—is taking place, because that is the only call that it is not involved in. The agents know that after three steps all agents know all secrets. In each step there is some change in common knowledge, that finally results in common knowledge of all secrets. For any call sequence of the Fixed Schedule this remains the case for more than four agents, under conditions of synchronicity and when we then assume that there is common knowledge which call sequence is executed, and with what time interval between calls - we could imagine the agents sitting around a table and making the calls from there, in view of each other, but whispering, so that any other person only notices that a call is made, but not what is said.

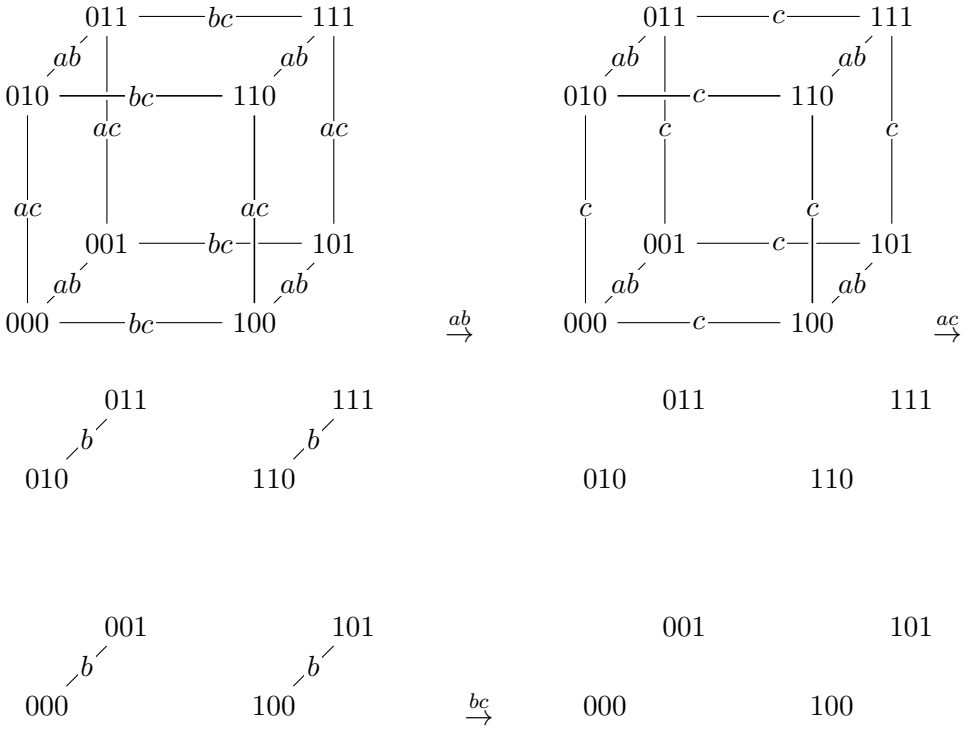



FIGURE 3.3: **Information transitions for the sequence of calls:** $ab; ac; bc$**.**

Now consider four agents, and a call between agent $a$ and agent $b$ such that agent $c$ and agent $d$ consider any other call possible that does not involve them. Although the real transition is $A.B.C.D \stackrel{ab}{\rightarrow} AB.AB.C.D$, agent $c$ considers it possible that the transition was $A.B.C.D \stackrel{ad}{\rightarrow} AD.A.C.AD$. As we are here only interested in what secrets are learnt from the call, we abstract from who initiates the call and who receives it, so $ab$ and $ba$ are treated on a par, for now. So we get the transitions shown in the right hand side of Figure 3.4. Let the transitions shown on the right hand side of Figure 3.4 be considered as a simple graphical depiction of a gossip model. Then, the unit of

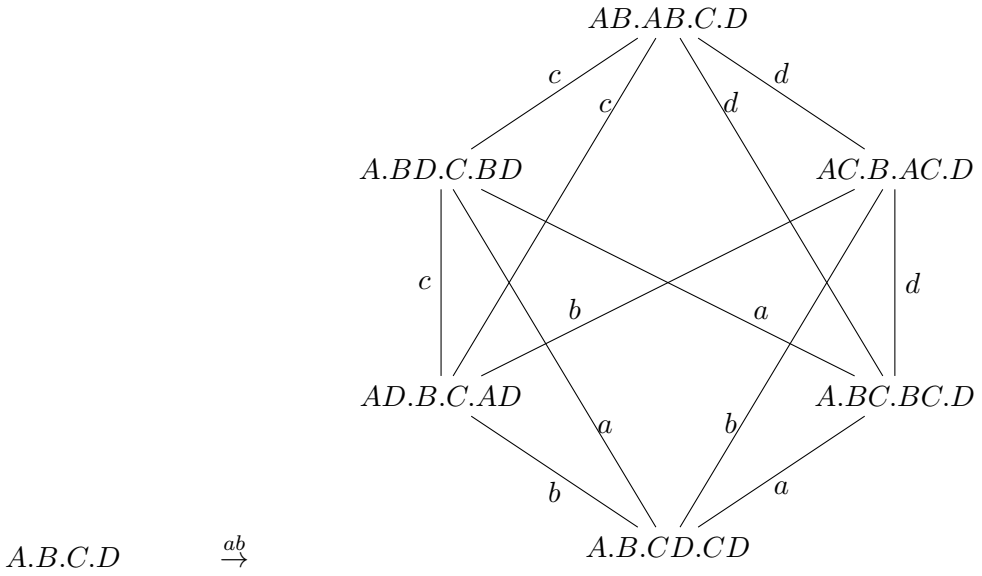



FIGURE 3.4: **A gossip model after a call at the initial situation.**

interpretation of a call is the combination of the gossip model and a designated gossip situation, namely the result $AB.AB.C.D$ of the $ab$ call at the initial situation. Such a gossip model also represents a multi-agent epistemic model, as we will shortly see.

**Postconditions of Protocol Execution** One outcome of executing a gossip protocol could be that all the agents have *general knowledge of all secrets*, that is, every agent $a_i$ knows the value of all secrets $A, B, \ldots$ —wherein we consider such secrets as propositional variables with value true or false. Suppose that it is indeed the case that after executing a gossip protocol the agents attain general knowledge of all secrets. Do they know more than that? This depends somewhat on further assumptions. If we suppose that the agents have common knowledge that the protocol is the Fixed Schedule, and if we suppose a synchronous system, then after the last call of each execution sequence all the agents have common knowledge that all secrets are general knowledge. This is because each of the agents knows that for any possible execution sequence of the Fixed Schedule, after $2n - 4$ calls all the agents know all the secrets.

For an epistemic gossip protocol, it may not be the case that the secrets are common knowledge after termination of the protocol. Take the Learn New Secrets protocol, and four agents, and assume also that the system is synchronous. The executions consist of between four and six calls (between $2n - 4$ and $n(n - 1)/2$). What if general knowledge is already obtained after four calls? The two agents not involved in that call do not necessarily know that all the secrets are now general knowledge. In this case, 'stuffing' the protocol with a number of *skip* actions will still achieve common knowledge, after six time steps.

Let us now proceed, in the next sections, to formalise the intuitions described so far.

## 3.4 Logical Dynamics of Gossip

### 3.4.1 Languages and Structures

Let a finite set of $n$ *agents* $\mathsf{Ag} = \{a, b, \dots\}$ and a corresponding set of *secrets* (propositional atoms) $\mathsf{P} = \{A, B, \dots\}$ be given. Recall (from Chapter 2) that upper case letters (for example, $A$) denote the secrets of the agents denoted by the corresponding lower case letters (for example, $a$).

**Definition 3.1** (Language $\mathcal{L}_K$)**.** We consider three modes $\mu \in \{-, 0, +\}$ of phone calls $ab^\mu$ (see below Section 3.5). We define $\mathcal{L}_K$ as:

$$\begin{array}{rrcl} \mathcal{L}_K \ni & \varphi & ::= & A \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \to \varphi) \mid K_a\varphi \mid [\pi]\varphi \\ & \pi & ::= & ?\varphi \mid ab^\mu \mid \mathsf{skip} \mid (\pi\,;\pi) \mid (\pi \cup \pi) \mid \pi^* \end{array}$$

where $a \neq b \in \mathsf{Ag}$, and $A \in \mathsf{P}$.

The program fragment $\pi$ of the language $\mathcal{L}_K$ adopts the usual Propositional Dynamic Logic constructs (see [9, Section 1.2]). The construct $[ab^\mu]\varphi$ stands for 'after a call of mode $\mu$ between agents $a$ and $b$, $\varphi$ (is true)'. For $(?\varphi\,;\pi)^*\,;?\neg\varphi$ we may write '`while` $\varphi$ `do` $\pi$'. Epistemic protocols will be defined as such programs $\pi$ but with additional constraints. Informally, a protocol is a program that intends to get all agents to know all secrets.

We also consider the languages $\mathcal{L}_{Kw}$ and $\mathcal{L}^*_{Kw}$. The language $\mathcal{L}_{Kw} \subseteq \mathcal{L}_K$ is such that the atomic formulas are of the form $Kw_aB$, which is an abbreviation for the $\mathcal{L}_K$ formula $K_aB \vee K_a\neg B$, meaning: '$a$ knows $b$'s secret', or '$a$ knows *whether* $B$'. The language $\mathcal{L}^*_{Kw} \subset \mathcal{L}_{Kw}$ uses only the static operators of $\mathcal{L}_{Kw}$. The definition of the languages $\mathcal{L}_{Kw}$ and $\mathcal{L}^*_{Kw}$ are given below.

**Definition 3.2** (Language $\mathcal{L}_{Kw}$)**.** The language $\mathcal{L}_{Kw}$ is defined as follows:

$$\begin{array}{rrcl} \mathcal{L}_{Kw} \ni & \varphi & ::= & Kw_aB \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \to \varphi) \mid K_a\varphi \mid [\pi]\varphi \\ & \pi & ::= & ?\varphi \mid ab^\mu \mid \mathsf{skip} \mid (\pi\,;\pi) \mid (\pi \cup \pi) \mid \pi^* \end{array}$$

**Definition 3.3** (Language $\mathcal{L}^*_{Kw}$)**.** The language $\mathcal{L}^*_{Kw}$ is defined as follows:

$$\mathcal{L}^*_{Kw} \ni \quad \varphi \quad ::= \quad Kw_aB \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \to \varphi) \mid K_a\varphi$$

Let us now present a set of definitions for the structures on which we will interpret epistemic formulas for our gossip scenarios. We begin by re-presenting the definition of epistemic model given in Chapter 2 as follows.

**Definition 3.4** (Epistemic model)**.** An epistemic model $M$ is a tuple $M = (S, \sim, V)$ such that:

- $S$ is a non-empty set of possible worlds,

- $\sim : \mathsf{Ag} \to \mathcal{P}(S \times S)$ assigns an equivalence relation to each agent,
- $V : S \to \mathsf{P} \to \{0,1\}$ is a valuation for each $s \in S$.

If $M = (S, \sim, V)$, rather than $s \in S$, we will also write $s \in M$. For $\sim(a)$ we write $\sim_a$, and for $V(s)$ we write $V_s$. A *pointed epistemic model* is a pair $(M, s)$ where $s \in M$. We also consider multi-pointed epistemic models $(M, S')$, where $S' \subseteq S$.

Epistemic models are also known as $S5$-models, and the $S5$ validities are well-known (see Section 2.1.1; for further references, see [22]). The scenarios we envisage will only use some specific $S5$-models.

**Definition 3.5** ($\equiv^M_{\mathsf{Q}}$)**.** Given an epistemic model $M = (S, \sim, V)$, and given some $\mathsf{Q} \subseteq \mathsf{P}$. Then, for every pair $s, t \in S$, $\quad s \equiv^M_{\mathsf{Q}} t$ if and only if: $V_s(A) = V_t(A)$ for all $A \in \mathsf{Q}$.

With respect to notation, note that when the gossip situation $M$ is clear from the context, we will sometimes write $\equiv_{\mathsf{Q}}$ rather than $\equiv^M_{\mathsf{Q}}$.

**Definition 3.6** (Gossip situation)**.** An epistemic model $M = (S, \sim, V)$ is a *gossip situation* if $S = \{s_M \mid s \in \{0,1\}^{|\mathsf{P}|}\}$ (the domain consists of all valuations[‡] ), and for every $a \in \mathsf{Ag}$, $\sim_a$ equals $\equiv_{\mathsf{Q}}$ for some $\mathsf{Q} \subseteq \mathsf{P}$ with $A \in \mathsf{Q}$. The *initial* (respectively *terminal*) gossip situation is the situation in which, for all agents $a$, $\mathsf{Q} = \{A\}$ (respectively, $\mathsf{Q} = \mathsf{P}$).

See Figure 3.1 for a graphical illustration of an initial gossip situation for three agents.

**Definition 3.7** (Gossip model)**.** A *gossip model* is a pair $G = (\mathcal{S}, \approx)$, where $\mathcal{S}$ is a set of gossip situations and $\approx$ assigns to each agent an equivalence relation $\approx_a$ on $\mathcal{S}$ satisfying, for all $M = (S, \sim^M, V)$ and $N = (T, \sim^N, W)$:

$$M \approx_a N \text{ iff } \exists \mathsf{Q} : \sim^M_a = \equiv^M_{\mathsf{Q}} \text{ and } \equiv^N_{\mathsf{Q}} = \sim^N_a \tag{3.1}$$

A *pointed gossip model* is a pair $(G, M) = ((\mathcal{S}, \approx), M)$, where $M \in G$. The *initial gossip model* is the (singleton) gossip model consisting of the initial gossip situation.

A gossip situation $M$ is a description of who knows which secret. In a gossip situation, if $\sim_a$ equals $\equiv_{\mathsf{Q}}$ for $\mathsf{Q} \subseteq \mathsf{P}$, this means that agent $a$ knows exactly the value of the secrets in $\mathsf{Q}$ (as in Theorem 3.8). An alternative way to represent a gossip situation $M$ is by a function $f_M : \mathsf{Ag} \to \mathcal{P}(\mathsf{P})$ where $f_M(a)$ denotes the secrets that are known by agent $a$. So: $M \models Kw_aD$ iff $D \in f_M(a)$. We may represent such a function as a list: $AB.ABC.ABC.D$ for instance is the function $f$ where $a$ knows the secrets $A$ and $B$, $b$ knows $A$, $B$ and $C$, etc.

We also can describe gossip models similarly: they can be represented as $\mathcal{F} = (F, \approx)$ where $F$ is a set of functions $\mathsf{Ag} \to \mathcal{P}(\mathsf{P})$ and $\approx_a$ an equivalence relation for every agent

[‡]Seeing states as abstract names we may still write $V_{s_M}$ for the valuation in $s_M$, although strictly $V_{s_M} = s$. Furthermore, when it is clear, from the context, to which gossip situation $s$ belongs we will omit the subscript $M$ and write $s$ rather $s_M$.

$a$, defined by $f \approx_a g$ iff $f(a) = g(a)$. An advantage of this functional representation is its succinctness: Figure 3.4 gives an example.

So, whereas a gossip situation encodes that for each agent $a$ there is a $\mathsf{Q}$ such that agent $a$ knows exactly the secrets in $\mathsf{Q}$, a gossip model allows agents to be uncertain which gossip situation is the actual one.

### Semantics of the Static Operators of $\mathcal{L}_K$ and $\mathcal{L}_{Kw}$

The truth conditions for the static operators of the language $\mathcal{L}_K$ on epistemic models are the same as those for the operators of the language $\mathcal{L}$ on epistemic models, as given in Section 2.1. We now present the truth conditions for the operators of the language $\mathcal{L}^*_{Kw}$ on gossip models, but before we do so we first present and prove a useful theorem. (Note that the static operators of $\mathcal{L}_{Kw}$ are the same as the operators of $\mathcal{L}^*_{Kw}$, therefore the truth conditions presented for the operators of $\mathcal{L}^*_{Kw}$ are the same for the static operators of $\mathcal{L}_{Kw}$, on gossip models).

**Theorem 3.8.** *Let $M = (S, \sim, V)$ be a gossip situation, and $s \in S, a \in \mathsf{Ag}, B \in \mathsf{P}$, and assume $\sim_a = \equiv^M_{\mathsf{Q}}$. Then*

$$M, s \models Kw_aB \text{ iff } B \in \mathsf{Q}$$

*and, consequently,*

*(i)* $M, s \models Kw_aB$ *iff* $M \models Kw_aB$;

*(ii)* $M, s \models \neg Kw_aB$ *iff* $M \models \neg Kw_aB$;

*(iii)* $M \models Kw_aB$ *or* $M \models \neg Kw_aB$

*Proof.* From left to right, suppose $M, s \models Kw_aB$, i.e., $M, s \models K_aB \vee K_a\neg B$. Now suppose towards contradiction that $B \notin \mathsf{Q}$. Consider the state $t$ such that $s \sim_a t$, and $V_s(X) \neq V_t(X)$ iff $X = B$ ($t$ only differs from $s$ in that it assigns a different value to $B$). Then $s \equiv^M_{\mathsf{Q}} t$, but then $M, s \models \neg K_aB \wedge \neg K_a\neg B$, contradicting our assumption. From right to left is immediate: if $B \in \mathsf{Q}$, then all states that $a$ considers possible in $s$ agree on the value they assign to $B$, so we have $M, s \models (B \to K_aB) \wedge (\neg B \to K_a\neg B)$. Since $M, s \models B \vee \neg B$, we then obtain $M, s \models K_aB \vee K_a\neg B$.

For $(i)$, from the foregoing, given $s \in S$, we obtain $M, s \models Kw_aB$ iff $B \in Q$ iff $M, t \models Kw_aB$, for any $t \in S$, from which we indeed obtain $M, s \models Kw_aB$ iff $\forall t\ M, t \models Kw_aB$ iff $M \models Kw_aB$. For $(ii)$, right to left is immediate as well, and for the other direction, note that $M, s \models \neg Kw_aB$ implies $B \notin Q$, and hence $M, t \models \neg Kw_aB$ for any $t$.

For $(iii)$, we have from $(i)$ that $M, s \models Kw_aB$ iff $M \models Kw_aB$ iff $B \in Q$. Also, from the main proposition and by contraposition, we have that $M, s \models \neg Kw_aB$ iff $B \notin Q$, and from $(ii)$ we have that $M, s \models \neg Kw_aB$ iff $M \models \neg Kw_aB$. Therefore $M \models \neg Kw_aB$ iff $B \notin Q$. Since either $B \in Q$ or $B \notin Q$, we then obtain $M \models Kw_aB$ or $M \models \neg Kw_aB$. □

The above theorem says that, if we are interested in who knows which secret, a gossip situation is an appropriate level of abstraction to reason about that. In particular, item

(*iii*) above indicates that a gossip situation, even if it is a collection of states (valuations), can be conceived of as a 'state' on itself. This item also justifies the first clause in the next truth definition.

**Definition 3.9** (Interpretation of the formulas of $\mathcal{L}^*_{Kw}$ on gossip models)**.** Let $(G, M)$ be a pointed gossip model, where $G = (\mathcal{S}, \approx)$. We inductively define the conditions under which a formula $\varphi \in \mathcal{L}^*_{Kw}$ holds in $(G, M)$, as follows:

$$\begin{array}{lll}
G, M \models_g Kw_aB & \text{iff} & M \models Kw_aB \\
G, M \models_g \neg\varphi & \text{iff} & G, M \not\models_g \varphi \\
G, M \models_g \varphi \wedge \psi & \text{iff} & G, M \models_g \varphi \text{ and } G, M \models_g \psi \\
G, M \models_g \varphi \vee \psi & \text{iff} & G, M \models_g \varphi \text{ or } G, M \models_g \psi \\
G, M \models_g \varphi \to \psi & \text{iff} & G, M \models_g \neg\varphi \text{ or } G, M \models_g \psi \\
G, M \models_g K_a\psi & \text{iff} & G, N \models_g \psi \quad \text{for every } N \text{ such that } M \approx_a N
\end{array}$$

If a formula $\varphi \in \mathcal{L}^*_{Kw}$ holds in $(G, M)$ for all $M \in \mathcal{S}$, then we write $G \models_g \varphi$. Alternatively, we say that $\varphi$ is valid in $G$. If a formula $\varphi \in \mathcal{L}^*_{Kw}$ is valid in all gossip models, we write $\models_g \varphi$.

**Notation.** Having defined the relation '$\models_g$' in the foregoing definition, we will however henceforth adopt the simpler notation '$\models$' when it is clear from the context that a given formula $\varphi$ is being interpreted on a gossip model. Specifically, we will write $G, M \models \varphi$ for $G, M \models_g \varphi$, and write $G \models \varphi$ for $G \models_g \varphi$ (likewise, we will write $G, M \not\models \varphi$ for $G, M \not\models_g \varphi$, and write $G \not\models \varphi$ for $G \not\models_g \varphi$), but we will retain the notation '$\models_g$' when we intend to express that a formula $\varphi$ is valid in all gossip models, in which case we will still write $\models_g \varphi$.

### 3.4.2 Logical Properties of Gossip Models and Gossip Situations

We now present and prove some of the properties of gossip models and gossip situations. We begin with some elementary validities on gossip situations as follows.

**Proposition 3.10.** *Let $G = (\mathcal{S}, \approx)$, and let $M, N \in \mathcal{S}$. Suppose $\sim_a^M = \equiv_{\mathsf{Q}}^M$ and $\sim_a^N = \equiv_{\mathsf{Q}}^N$, for some $\mathsf{Q} \subseteq \mathsf{P}$ and for $a \in \mathsf{Ag}$. Then*

$$G, M \models Kw_aB \quad \textit{if and only if} \quad G, N \models Kw_aB$$

*Proof.*

$$\begin{array}{rcll}
G, M \models Kw_aB & \iff & M \models Kw_aB & \text{(Definition 3.9)} \\
M \models Kw_aB & \iff & B \in \mathsf{Q} & \text{(Theorem 3.8)}
\end{array}$$

Likewise,

$$\begin{array}{rcl}
G, N \models Kw_aB & \iff & N \models Kw_aB \\
N \models Kw_aB & \iff & B \in \mathsf{Q}
\end{array}$$

Therefore,

$$G, M \models Kw_aB \iff B \in \mathsf{Q} \iff G, N \models Kw_aB$$

So we conclude that:

$$G, M \models Kw_aB \iff G, N \models Kw_aB \qquad \square$$

**Corollary 3.11.** *Let $G = (\mathcal{S}, \approx)$ and $M, N \in \mathcal{S}$. Suppose $M \approx_a N$, for some $a \in \mathsf{Ag}$. Then:*

$$G, M \models Kw_aB \text{ if and only if } G, N \models Kw_aB$$

*Proof.* The proof of Corollary 3.11 follows from Definition 3.7 and Proposition 3.10. $\square$

Corollary 3.11 tells us that if an agent considers two gossip situations to be equivalent, then it knows the same secrets in both gossip situations.

**Proposition 3.12.** *If $M$ is a gossip situation and $\varphi \in \mathcal{L}^*_{Kw}$, then $M \models \varphi$ or $M \models \neg\varphi$.*

*Proof.* We need to show that if a formula $\varphi \in \mathcal{L}^*_{Kw}$ is not valid in a gossip situation, then $\neg\varphi$ is valid in that situation. Now let $M = (T, \sim, V)$ be any gossip situation. We proceed by induction on $\varphi \in \mathcal{L}^*_{Kw}$.

**Base Case: $\varphi$ is $Kw_aB$.** This follows from (the proof of) Theorem 3.8, that is, we have that:

$$M \models Kw_aB \text{ or } M \models \neg Kw_aB$$

**Inductive Hypothesis** For every $\varphi' \in \mathcal{L}^*_{Kw}$, it is the case that:

$$M \models \varphi' \text{ or } M \models \neg\varphi'$$

**Inductive Step** Let $\varphi', \varphi'' \in \mathcal{L}^*_{Kw}$. We distinguish the following cases.

**Case $\varphi$ is $\neg\varphi'$.** This case is straightforward from the inductive hypothesis:

$$M \models \varphi' \text{ or } M \models \neg\varphi'$$

**Case $\varphi$ is $(\varphi' \wedge \varphi'')$.** By the inductive hypothesis:

$$(M \models \varphi' \text{ or } M \models \neg\varphi') \text{ and } (M \models \varphi'' \text{ or } M \models \neg\varphi'')$$

We will consider then four possible cases as a result of the inductive hypothesis: (i) $M \models \varphi'$ and $M \models \varphi''$, (ii) $M \models \varphi'$ and $M \models \neg\varphi''$, (iii) $M \models \neg\varphi'$ and $M \models \varphi''$, and (iv) $M \models \neg\varphi'$ and $M \models \neg\varphi''$.

From propositional logic, cases (i)-(iv) can be written correspondingly as follows: (i) $M \models \varphi' \wedge \varphi''$ (ii) $M \models \varphi' \wedge \neg\varphi''$ (iii) $M \models \neg\varphi' \wedge \varphi''$ (iv) $M \models \neg\varphi' \wedge \neg\varphi''$.

For Case (i), obviously, $M \models \varphi' \wedge \varphi''$ if and only if $M \models \varphi$. And, each of the cases (ii), (iii) and (iv) implies that $M \models \neg(\varphi' \wedge \varphi'')$, which is equivalent to $M \models \neg\varphi$ (by the assumption on $\varphi$).

**Case $\varphi$ is $K_a\varphi'$.** By the inductive hypothesis, $M \models \varphi'$ or $M \models \neg\varphi'$. Now, since only one of $\varphi$ and $\neg\varphi'$ can hold in the same $M$, let us consider the following two cases. First, suppose that it is the case that $M \models \varphi'$. Then for every $s \in M$, we have that $M, s \models \varphi'$ holds. But then it also holds that for every $s, t \in M$, $s \sim_a t$ implies $M, t \models \varphi'$, which is equivalent to $M, s \models K_a\varphi'$ (from the semantics of the '$K_a$' operator). And since $M, s \models K_a\varphi'$ for all $s \in M$, we conclude that $M \models K_a\varphi'$ holds, which is equivalent to $M \models \varphi$ (by the assumption on $\varphi$).

For the second case, suppose that $M \models \neg\varphi'$. Then, similar to the foregoing argument, $M \models K_a\neg\varphi'$, which implies that $M \models \neg K_a\varphi'$ (from the semantics of the '$K_a$' operator). And so we conclude that $M \models \neg\varphi$ (by the assumption on $\varphi$).

This concludes the inductive argument. □

**Proposition 3.13.** *Let $M = (S, \sim, V)$ be a gossip situation, and let $\varphi \in \mathcal{L}^*_{Kw}$. Then: $M \models K_a\varphi \to K_bK_a\varphi$, for $a \in \mathsf{Ag}$.*

Proposition 3.13 says that in a gossip situation all knowledge is public. We give the proof of this proposition below.

*Proof of Proposition 3.13.* From Proposition 3.12 we know that $M \models K_a\varphi$ or $M \models \neg K_a\varphi$. Now suppose $M \models K_a\varphi$. That is $M, t \models K_a\varphi$, for all $t \in M$. Now take $s \in M$, and any $u$ such that $s \sim_b u$. Since $M, u \models K_a\varphi$, we have that $M, s \models K_bK_a\varphi$ (from the semantics of '$K_b$' operator).

On the other hand, suppose $M \models \neg K_a\varphi$. We then obtain by propositional logic that $M \models K_a\varphi \to K_bK_a\varphi$. □

**Proposition 3.14.** *A gossip model is an epistemic model.*

*Proof.* Each gossip model $G$ gives rise to an epistemic model $E(G) = (J, \sim, X)$ where:

- $J = \{s_M \mid M = (S, \sim, V) \in \mathcal{S} \text{ and } s \in S\}$;
- $s_M \sim_a t_N$ with $M = (S, \sim^M, V), N = (T, \sim^N, U)$ iff there are $v_M$ and $u_N$ such that $V_v = U_u$, $s \sim^M_a v$, $t \sim^N_a u$, and $M \approx_a N$ (†)
- $X_{s_M} = V_s$ (††)

We show the following:

1. If $s_M \sim_a t_N$, then there is a set of atoms $\mathsf{Q} \subseteq \mathsf{P}$ such that $X_{s_M}(A) = X_{t_N}(A)$ for all $A \in \mathsf{Q}$.
2. If $s \sim^M_a v$ then $s_M \sim_a v_M$

3. $\sim_a$ is an equivalence relation.

1. Take $s_M, t_N \in J$ with $s_M \sim_a t_N$. It implies that there are $v_M$ and $u_N$ such that $V_v = U_u, s \sim_a^M v, t \sim_a^N u$ and $M \approx_a N$. The later implies that there is some $\mathsf{Q} \subseteq \mathsf{P}$ such that $\sim_a^M = \equiv_{\mathsf{Q}}^M$ and $\equiv_{\mathsf{Q}}^N = \sim_a^N$ (from Definition 3.7). Now, let $A \in \mathsf{Q}$. We have:

$$\begin{array}{rcll} X_{s_M}(A) & = & V_s(A) & \text{(by (††))} \\ & = & V_v(A) & \text{(since } \sim_a^M = \equiv_{\mathsf{Q}}^M \text{ and } s \sim_a^M v\text{)} \\ & = & U_u(A) & \text{(since } V_v = U_u\text{)} \\ & = & U_t(A) & \text{(since } \sim_a^N = \equiv_{\mathsf{Q}}^N \text{ and } u \sim_a^N t\text{)} \\ & = & X_{t_N}(A) & \text{(by (††))} \end{array}$$

2. We now show that $s \sim_a^M v$ implies $s_M \sim_a v_M$. This follows directly from the definition of $\sim_a$: take $M = N$, $t = u = v \in S$, then we get $V_t = U_u$, $s \sim_a^M t$, $v \sim_a^N u$ (from the reflexivity of $\sim_a^N$), and $M \approx_a N$, and hence, by (†), we have $s_M \sim_a v_M$.

3. Reflexivity of $\sim_a$ follows directly from Item 2: from $s \sim_a^M s$, we obtain $s_M \sim_a s_M$. From Item 1 it is also clear that $\sim_a$ is symmetric. For transitivity, suppose for $M = \langle S, \sim^M, V\rangle$, $N = \langle T, \sim^N, U\rangle$ and $O = \langle W, \sim^O, Z\rangle$ we have $s_M \sim_a t_N$ and $t_N \sim_a w_O$. This gives us:

$$\begin{array}{l} \exists s'_M, t'_N \text{ s.t. } (i)\ V_{s'} = U_{t'}, (ii)\ s \sim_a^M s', (iii)\ t \sim_a^N t', \text{ and } (iv)\ M \approx_a N, \text{ and} \\ \quad \exists \tau'_N, \omega'_O \text{ s.t. } (v)\ U_{\tau'} = Z_{\omega'}, (vi)\ t \sim_a^N \tau', (vii)\ w \sim_a^O \omega', \text{ and } (viii)\ N \approx_a O \end{array}$$

We need to show that we also have that $s_M \sim_a w_O$. Since by definition, a gossip situation contains all possible valuations for the secrets, there is a $w' \in W$ for which $V_{s'} = Z_{w'}$. Note that, by $(iv)$ and $(viii)$ and by Definition 3.7, there is some $\mathsf{Q} \subseteq \mathsf{P}$ such that in each of the three gossip situations, two points are indistinguishable for $a$ iff they agree on the valuation of atoms in $\mathsf{Q}$ (in particular, we obtain $M \approx_a O$). Take an arbitrary $A \in \mathsf{Q}$. Then obviously $V_{s'}(A) = Z_{w'}(A)$. Moreover,

$$\begin{array}{rcll} V_s(A) & = & V_{s'}(A) & \text{(since } s \sim_a^M s'\text{)} \\ & = & U_{t'}(A) & \text{(by } (i)\text{)} \\ & = & U_t(A) & \text{(by } (iii)\text{)} \\ & = & U_{\tau'}(A) & \text{(by } (vi)\text{)} \\ & = & Z_{\omega'}(A) & \text{(by } (v)\text{)} \\ & = & Z_w(A) & \text{(by } (vii)\end{array}$$

We conclude that for all $A \in \mathsf{Q}$, $Z_w(A) = Z_{w'}(A)$. In sum, we now have:

$$V_{s'} = Z_{w'}, s \sim_a^M s', w \sim_a^O w', \text{ and } M \approx_a O,$$

which allows us to conclude $s_M \sim_a w_O$. □

The pointed gossip model $(G, M)$ corresponds to the multi-pointed epistemic model $(E(G), S)$, where $S$ is the domain of $M$.

**Proposition 3.15.** *Let $(G, M)$ be a gossip model and $\varphi \in \mathcal{L}^*_{Kw}$. Then*

$$G, M \models \varphi \iff E(G), S \models \varphi$$

*where $E(G) = (J, \sim, X)$ is as in Proposition 3.14, $G = (\mathcal{S}, \approx)$, $M = (S, \sim^M, V)$, and $S$ is equal to the domain of $M$.*

*Proof.* Let $s_M, t_M \in S$, where $M \in \mathcal{S}$. We begin with proofs of some ancillary items, as follows:

1. $s_M \sim_a t_M$ if and only if $s \sim_a^M t$
2. $E(G), S \models Kw_aB$ if and only if $E(G), s_M \models Kw_aB$ if and only if $M \models Kw_aB$
3. $E(G), S \models \neg Kw_aB$ if and only if $E(G), s_M \models \neg Kw_aB$ if and only if $M \models \neg Kw_aB$
4. $E(G), S \models Kw_aB$ or $E(G), S \models \neg Kw_aB$

The proofs of Items 1 through 4 are as follows.

1. We first show the following:

   - $E(G), s_M \models Kw_aB$ if and only if $X_{s_M}(B) = X_{t_M}(B)$ for all $t_M$ such that $s_M \sim_a t_M$
   - $M, s \models Kw_aB$ if and only if $V_s(B) = V_t(B)$ for all $t$ such that $s \sim_a^M t$

$$\begin{array}{rcll}
E(G), s_M \models Kw_aB & \iff & E(G), s_M \models K_aB \vee K_a\neg B & (\Diamond) \\
& \iff & E(G), s_M \models K_aB & \\
& \text{or} & E(G), s_M \models K_a\neg B & \\
& \iff & E(G), t_M \models B & \forall t_M \text{ such that } s_M \sim_a t_M \\
& \text{or} & E(G), t_M \models \neg B & \forall t_M \text{ such that } s_M \sim_a t_M \\
& \iff & X_{t_M}(B) = 1 & \forall t_M \text{ such that } s_M \sim_a t_M \\
& \text{or} & X_{t_M}(B) = 0 & \forall t_M \text{ such that } s_M \sim_a t_M \\
& \iff & X_{s_M}(B) = X_{t_M}(B) & \forall t_M \text{ such that } s_M \sim_a t_M
\end{array}$$

   Similarly,

$$\begin{array}{rcll}
M, s \models Kw_aB & \iff & M, s \models K_aB \vee K_a\neg B & (\Diamond\Diamond) \\
& \iff & M, s \models K_aB & \\
& \text{or} & M, s \models K_a\neg B & \\
& \iff & M, t \models B & \forall t \text{ such that } s \sim_a^M t \\
& \text{or} & M, t \models \neg B & \forall t \text{ such that } s \sim_a^M t \\
& \iff & V_t(B) = 1 & \forall t \text{ such that } s \sim_a^M t \\
& \text{or} & V_t(B) = 0 & \forall t \text{ such that } s \sim_a^M t \\
& \iff & V_s(B) = V_t(B) & \forall t \text{ such that } s \sim_a^M t
\end{array}$$

We now come to the left-to-right proof of Item 1. Suppose $s_M \sim_a t_M$. From the definition of $\sim_a$ given in Proposition 3.14, we obtain that:

> $s_M \sim_a t_M$ if and only if there exists some $v \in M$ and some $u \in M$ such that $V_v = V_u$, $s \sim_a^M v$ and $t \sim_a^M u$ and $M \approx_a M$ (†)

Since the domain of $M$ is the set of all valuations for the atoms in $\mathsf{P}$, then $V_v = V_u$ implies $v = u$. Therefore we have that $s \sim_a^M v$ and $t \sim_a^M u$ imply $s \sim_a^M t$ (recall that $\sim_a^M$ is an equivalence relation).

For the right-to-left proof of Item 1: $s \sim_a^M t$ implies $s_M \sim_a t_M$ follows directly from the definition of $\sim_a$: suppose $s \sim_a^M t$, then from (†) take $v = t = u \in M$, then we get $V_v = V_u$, $s \sim_a^M v$, $t \sim_a^M u$ and $M \approx_a M$, and hence we have $s_M \sim_a t_M$.

2. From ($\Diamond$), ($\Diamond\Diamond$) and Item 1, and given that $X_{s_M} = V_s$ for all $s_M \in S$, we conclude that:

$$\begin{array}{rcll} E(G), s_M \models Kw_aB & \iff & M, s \models Kw_aB & \\ & \iff & M \models Kw_aB & \text{(from Theorem 3.8)} \\ & \iff & M, s \models Kw_aB & \text{for every } s \in S \\ & \iff & E(G), s_M \models Kw_aB & \text{for every } s_M \in S \\ & \iff & E(G), S \models Kw_aB & \end{array} \quad (\ddagger)$$

3. Similar to Item 2, we obtain the following:

$$\begin{array}{rcll} E(G), s_M \models \neg Kw_aB & \iff & M, s \models \neg Kw_aB & \text{(contraposition on (‡), truth conditions for '¬')} \\ & \iff & M \models \neg Kw_aB & \text{(from Theorem 3.8)} \\ & \iff & M, s \models \neg Kw_aB & \text{for every } s \in S \\ & \iff & E(G), s_M \models \neg Kw_aB & \text{for every } s_M \in S \\ & \iff & E(G), S \models \neg Kw_aB & \end{array} \quad (\ddagger\ddagger)$$

4. From Item 2 and Item 3 we have, respectively, $M \models Kw_aB \iff E(G), S \models Kw_aB$ and $M \models \neg Kw_aB \iff E(G), S \models \neg Kw_aB$. But from Theorem 3.8 we know that $M \models Kw_aB$ or $M \models \neg Kw_aB$. Hence we see also that $E(G), S \models Kw_aB$ or $E(G), S \models \neg Kw_aB$.

To prove the main proposition, we now proceed with the base cases of the inductive proof on $\varphi, \psi \in \mathcal{L}^*_{Kw}$.

**Base Case 1: $\varphi$ is $Kw_aB$.**

$$\begin{array}{rcll} G, M \models Kw_aB & \iff & M \models Kw_aB & \text{(from Definition 3.9)} \\ & \iff & E(G), S \models Kw_aB & \text{(from (‡))} \end{array}$$

**Base Case 2: $\varphi$ is $\neg Kw_aB$.**

$$\begin{aligned} G, M \models \neg Kw_aB &\iff M \models \neg Kw_aB && \text{(from Definition 3.9)} \\ &\iff E(G), S \models \neg Kw_aB && \text{(from (‡‡))} \end{aligned}$$

**Inductive Hypothesis** For every $\varphi' \in \mathcal{L}^*_{Kw}$, it is the case that:

1. $G, M \models \varphi' \iff E(G), S \models \varphi'$
2. $G, M \models \neg\varphi' \iff E(G), S \models \neg\varphi'$

**Inductive Step** Let $\varphi', \varphi'' \in \mathcal{L}^*_{Kw}$. We distinguish the following cases.

**Case $\varphi$ is $\neg\varphi'$.** This case is immediate from the inductive hypothesis.

**Case $\varphi$ is $(\varphi' \wedge \varphi'')$.** By the inductive hypothesis:

$$G, M \models \varphi' \iff E(G), S \models \varphi' \quad \text{and} \quad G, M \models \varphi'' \iff E(G), S \models \varphi''.$$

Therefore, $G, M \models (\varphi' \wedge \varphi'') \iff E(G), S \models (\varphi' \wedge \varphi'')$ (by propositional logic).

**Case $\varphi$ is $K_a\varphi'$.** For the left-to-right case of the proposition, suppose that $G, M \models K_a\varphi'$. Now,

$$\begin{aligned} G, M \models K_a\varphi' &\iff G, N \models \varphi' && \text{for every } N \in \mathcal{S} \text{ such that } M \approx_a N \\ & && \text{(Definition 3.9)} \\ &\iff E(G), T \models \varphi' && \text{for every } N \in \mathcal{S} \text{ such that } M \approx_a N, \\ & && \text{where } T \text{ is the domain of } N \\ & && \text{(from the inductive hypothesis)} \\ &\iff E(G), s_N \models \varphi' && \text{for every } N \in \mathcal{S} \text{ such that } M \approx_a N, \\ & && \text{and for every } s_N \in T \text{ where } T \text{ is the} \\ & && \text{domain of } N \\ & && \text{(‡‡‡)} \end{aligned}$$

Now, let $\mathcal{X} = \{s \mid s \in N' \in \mathcal{S}, \text{ and } M \approx_a N'\}$ and $\mathcal{Y} = \{t \mid s \sim_a t, \text{ where } s \in M \text{ and } t \in N' \in \mathcal{S}\}$, then we see that $\mathcal{Y} \subseteq \mathcal{X}$ (the reason is from the definition of $\sim_a$ in Proposition 3.14 in which we have that whereas $M \approx_a N$ is a necessary condition for $s \sim_a t$, it is not a sufficient condition for $s \sim_a t$). But from (‡‡‡) we have that for every $s_N$ such that $s_N \in \mathcal{X}$, it is the case that $E(G), s_N \models \varphi'$. So, from the foregoing, it follows that for every $s_N$ such that $s_N \in \mathcal{Y}$, it is also the case that $E(G), s_N \models \varphi'$. That is, $E(G), s_N \models \varphi'$ for every $s_M \in S$ and for every $s_N \in J$ such that $s_M \sim_a s_N$. And from the semantics of '$K_a$' on epistemic models (see Definition 2.4) we therefore obtain:

$$\text{for every } s_M \in S, \quad E(G), s_M \models K_a\varphi' \iff E(G), S \models K_a\varphi'$$

On the other hand, for the right-to-left case of the proposition, suppose that $E(G), S \models K_a\varphi'$, where $M = (S, \sim^M, V)$. Now,

$$\begin{aligned} E(G), S \models K_a\varphi' &\iff E(G), s_M \models K_a\varphi' && \text{for every } s_M \in S \\ &\iff E(G), t_N \models \varphi' && \text{for every } s_M \in S, \text{ and for every} \\ & && t_N \in J \text{ such that } s_M \sim_a t_N \\ & && \text{(Definition 2.4)} \end{aligned}$$

(†††)

Now consider an arbitrary $N = (T, \sim^N, U)$ such that $M \approx_a N$. For each $s_M \in S$, there is a unique $s_N \in T$ such that $V_{s_M} = U_{s_N}$ (from the definition of $E(G)$ in Proposition 3.14; note also that from the same definition and for such $s_N$ we have also that $X_{s_M} = X_{s_N}$ since $X = V = U$), and for such $s_N$ we also have that $s_M \sim_a s_N$ (the reason is that since $X_{s_M} = X_{s_N}$ then there is some $v \in S$ (namely, $s_M$) and some $u \in T$ (namely, $s_N$) such that $s_M \sim_a^M v$, $s_N \sim_a^N u$ and $V_v = U_u$; therefore since we have that $M \approx_a N$, then we obtain $s_M \sim_a s_N$ (see the definition of $E(G)$ in Proposition 3.14)).

Furthermore, since the set of valuations of all the worlds in any gossip situation is equal to the set of all valuations for the atoms in $\mathsf{P}$, where $\mathsf{P}$ is the set of all secrets in the scenario, and since $|S| = |T| = 2^{|P|}$, then we have that for every world $t'_N \in T$, there is a unique world $s_M \in S$ such that $U_{t'_N} = V_{s_M} = X_{t'_N} = X_{s_M}$ and $s_M \sim_a t'_N$. And since for all $t_N \in J$ (for any $N$) such that $s_M \sim_a t_N$ we have that $E(G), t_N \models \varphi'$ (from (†††)), then we also have that for all $t'_N \in T$ that $E(G), t'_N \models \varphi'$. That is, $E(G), T \models \varphi'$.

Since in the foregoing we considered an arbitrary $N$ such that $M \approx_a N$, we then also obtain that $E(G), T \models \varphi'$ for all $N$ such that $M \approx_a N$, where $T$ is the domain of $N$. But from the inductive hypothesis, this is equivalent to saying that $G, N \models \varphi'$ for an arbitrary $N$ such that $M \approx_a N$, and from the semantics of '$K_a$' on gossip models (see Definition 3.9), this is equivalent to $G, M \models K_a\varphi'$.

From the foregoing left-to-right and right-to-left cases, we then conclude that: $G, M \models K_a\varphi'$ if and only if $E(G), S \models K_a\varphi'$, which then concludes the inductive argument.

□

## 3.5 Semantics of Calls and Protocols

We now proceed to define the interpretation of calls between two agents $a$ and $b$, and the interpretation of protocols consisting of such calls. We first consider calls, and then, protocols. We distinguish three *modes* of calls, $ab^-$ (public, synchronous), $ab^0$ (private, synchronous) and $ab^+$ (private, asynchronous). Given such a call, the agents $a$ and $b$ are the *callers* and all other agents are the *non-callers*. Note that we always assume that any such callers $a$ and $b$ are two different agents.

The $ab^-$ call models a telephone call in the 'traditional' network systems setting of gossiping protocols [30]: it is common knowledge between all agents that $a$ and $b$ are making a call, but the non-callers may not know the value of the secrets the callers exchange. We could say that all agents are sitting in a circle round a table, so the non-callers can observe the callers, but we imagine the callers to talk softly, so that the non-callers cannot hear what the callers say. They only know that the callers exchange all secrets they know.

The $ab^0$ call models a telephone call between $a$ and $b$ where the non-callers may not know who are making a call. But they know that a call is made. (The system is synchronised.) For example, given four agents $a, b, c, d$, when $a$ and $b$ are making a call, then $c$ considers it possible that the call was between $a$ and $d$, or between $b$ and $d$; $c$ only knows that it was not involved in the call itself.

The $ab^+$ call is like $ab^0$ but with the additional option that the non-callers consider it possible that no call took place at all (the $\mathsf{skip}$ action).

**Definition 3.16** (Call in a gossip situation)**.** Let $M = (S, \sim, V)$ be a gossip situation and $\psi \in \mathcal{L}_{Kw}$. Then:

$$M \models [ab^\mu]\psi \quad \text{iff} \quad M^{ab} \models \psi$$
$$M \models [\mathsf{skip}]\psi \quad \text{iff} \quad M \models \psi$$

where $\mu \in \{-, 0, +\}$ and $M^{ab} = (S, \sim', V)$ such that $\sim'_a = \sim'_b = \sim_a \cap \sim_b$, and for all $c \neq a, b$, $\sim'_c = \sim_c$.

The action of calling has no precondition. Two agents can always make a call. Their distributed factual knowledge thus becomes shared between the two. This is the intersection of $\sim_a$ and $\sim_b$ in the definition. The mode $\mu$ of a call is irrelevant for its interpretation in a gossip situation. The $\mathsf{skip}$ action has no informative or other consequences.

**Lemma 3.17.** *Let $M = (S, \sim, V)$ be a gossip situation. Suppose $\sim_a = \equiv_{\mathsf{Q}_1}$ and $\sim_b = \equiv_{\mathsf{Q}_2}$ then $\sim_a \cap \sim_b = \equiv_{\mathsf{Q}_1 \cup \mathsf{Q}_2}$.*

*Proof.* Suppose $\sim_a = \equiv_{\mathsf{Q}_1}$ and $\sim_b = \equiv_{\mathsf{Q}_2}$, where $a, b \in \mathsf{Ag}$. That is, $s \equiv_{\mathsf{Q}_1} t$ if and only if $V_s(A) = V_t(A)$ for all $A \in \mathsf{Q}_1$; and $s \equiv_{\mathsf{Q}_2} t$ if and only if $V_s(A') = V_t(A')$, for all $A' \in \mathsf{Q}_2$. Now let $\sim_a \cap \sim_b = \equiv_{\mathsf{Q}}$, then $s \equiv_{\mathsf{Q}} t$ if and only if $V_s(A) = V_t(A)$ and $V_s(A') = V_t(A')$, for all $A \in \mathsf{Q}_1$ and for all $A' \in \mathsf{Q}_2$. But this is equivalent to saying that $V_s(C) = V_t(C)$, for all $C \in \mathsf{Q}_1 \cup \mathsf{Q}_2$. We therefore conclude that $\mathsf{Q} = \mathsf{Q}_1 \cup \mathsf{Q}_2$. □

**Definition 3.18** (Semantics of calls in a gossip model)**.** Let $(G, M)$ be given, where $G = (\mathcal{S}, \approx)$ and $M \in \mathcal{S}$. We define $[\![\mathsf{skip}]\!] = \{((G, M), (G, M))\}$ and, for the modes of call $ab^\mu$, with $\mu \in \{-, 0, +\}$:

$$[\![ab^\mu]\!] \quad = \quad \{((G, M), (G^{\mathsf{call}_\mu}, M^{ab}))\}$$

where (for all modes $\mu$) $G^{\mathsf{call}\mu} = (\mathcal{S}^\mu, \approx^\mu)$, such that

$$\begin{array}{lcl} \mathcal{S}^- & = & \{N^{ab} \mid N \in \mathcal{S}\} \\ \mathcal{S}^0 & = & \{N^{cd} \mid N \in \mathcal{S} \text{ and } c \neq d \in \mathsf{Ag}\} \\ \mathcal{S}^+ & = & \mathcal{S}^0 \cup \mathcal{S} \end{array}$$

and (see Definition 3.7) for any $N, N' \in \mathcal{S}^\mu$: $N \approx^\mu_a N'$ if and only if there is a $\mathsf{Q} \subseteq \mathsf{P}$ such that $\sim^N_a = \equiv^N_{\mathsf{Q}}$ and $\equiv^{N'}_{\mathsf{Q}} = \sim^{N'}_a$. For the actions $\alpha \in \{\mathsf{skip}, \mathsf{call}^-, \mathsf{call}^0, \mathsf{call}^+\}$, we then define $G, M \models [\alpha]\varphi$ if and only if for all $((G, M), (G', M')) \in [\![\alpha]\!]$, $(G', M') \models \varphi$.

**Definition 3.19** (Interpreting complex programs)**.** The interpretation of calls on pointed gossip models $(G, M)$ of Definition 3.18 is lifted to arbitrary programs $\pi$ in a standard way, where again we take into account that for all $\varphi \in \mathcal{L}_{Kw}$, either $\varphi$ or $\neg\varphi$ is a model validity on a gossip situation: $M \models [\pi]\psi$ if and only if for all $M'$ such that $M[\![\pi]\!]M'$, $M' \models \psi$.

$$\begin{array}{lcll} [\![?\varphi]\!] & = & \{((G, M)(G, M))\} & \text{iff } \ G, M \models \varphi \\ [\![?\varphi]\!] & = & \emptyset & \text{iff } \ G, M \models \neg\varphi \\ [\![\pi; \pi']\!] & = & [\![\pi]\!] \cdot [\![\pi']\!] & \\ [\![\pi \cup \pi']\!] & = & [\![\pi]\!] \cup [\![\pi']\!] & \\ [\![\pi^*]\!] & = & [\![\pi]\!]^* & \end{array}$$

As a result of a call $ab^-$, for each existing gossip situation in a gossip model we get exactly one new gossip situation, namely the one in which $a$ and $b$ have exchanged their information. A call $ab^0$ has as a result that we have to consider the execution of any call between two agents in every gossip situation, not only the call between $a$ and $b$ in the actual gossip situation $M$. Thus, given $n$ agents, the number of gossip situations in the resulting gossip model due to a call is $n(n-1)$ times the number of gossip situations in the gossip model before the call, with each of the resulting gossip situations being the result of one particular call between two agents (note that we distinguish, for example, between the call $ab^0$ and the call $ba^0$). For $ab^+$ we also need to take the gossip situation in which nothing happened into account.

### 3.5.1 Logical Properties of a Call in a Gossip Model

We now present some logical properties due to a call in a gossip model.

**Proposition 3.20.** *Let $a, b, c \in \mathsf{Ag}$, and $D \in \mathsf{P}$. Let:*

$$\begin{array}{lcl} \mathcal{L}^\beta_{Kw} \ni \varphi & ::= & \beta \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid K_a\varphi \\ \mathcal{L}_\beta \ni \beta & ::= & Kw_aB \mid \neg\beta \mid (\beta \wedge \beta) \end{array}$$

*(Recall the equivalences $\beta_1 \vee \beta_2 \Leftrightarrow \neg(\neg\beta_1 \wedge \neg\beta_2)$ and $\beta_1 \to \beta_2 \Leftrightarrow \neg(\beta_1 \wedge \neg\beta_2)$.)*
*Let $\varphi \in \mathcal{L}^\beta_{Kw}$, and let $\mu, \mu' \in \{-, 0, +\}$. Then:*

1. $G \models [ab^\mu]Kw_cD \leftrightarrow [ab^{\mu'}]Kw_cD$

2. $G \models [ab^0]\varphi \to [ab^-]\varphi$

3. $G \models [ab^+]\varphi \to [ab^-]\varphi$

4. $G \models [ab^-]\varphi \leftrightarrow [ab^0]\varphi$ *if* $|\mathsf{Ag}| = 3$

Proposition 3.20 does not apply to general epistemic postconditions $K_c\psi$; we have for instance that $[ab^0]K_c\psi \to [ab^-]K_c\psi$ is *not* valid on gossip models: take $\psi = \neg K_c K w_b A$. The first item of this proposition says that the secrets one knows do not depend on the mode of call. None of the reversed implications of 1 and 2 of Proposition 3.20 is valid on gossip models. For example, take a scenario with four agents $a, b, c, d$ with respective secrets $A, B, C, D$. Suppose an $ab^-$ call is made at the initial gossip situation. After such an $ab^-$ call, agent $c$ knows that agent $a$ knows secret $B$. But after an $ab^0$ call at the initial gossip situation, agent $c$ considers it possible that agent $a$ does not know secret $B$ since agent $c$ considers it possible that the call may have been, for example, between agent $a$ and agent $d$. The same example also demonstrates that the reverse implication of Proposition 3.20(2) is not valid on gossip models.

We first prove the following general result.

**Theorem 3.21.** *Let $G = (\mathcal{S}, \approx)$ and $H = (\mathcal{S}', \approx')$ be gossip models, such that $M \in \mathcal{S}' \subseteq \mathcal{S}$. Then*

$$\forall \varphi \in \mathcal{L}^{\beta}_{Kw}\ G, M \models \varphi \Rightarrow H, M \models \varphi$$

*Proof.* First of all, note that $\approx'_a = \approx_a \cap (\mathcal{S}' \times \mathcal{S}')$: Suppose $M, N \in \mathcal{S}'$, then, by definition,

$$M \approx'_a N \text{ iff } \exists \mathsf{Q} \subseteq \mathsf{P}(\sim^M_a = \equiv^M_{\mathsf{Q}} \text{ and } \sim^N_a = \equiv^N_{\mathsf{Q}}) \text{ iff } M \approx_a N$$

We now prove the theorem with induction on $\varphi \in \mathcal{L}^{\beta}_{Kw}$, starting with formulas $\beta \in \mathcal{L}_{\beta}$. For such formulas, we even prove the stronger

$$G, M \models \beta \Leftrightarrow H, M \models \beta \tag{3.2}$$

First, suppose $\beta = Kw_aB$. Then: $G, M \models Kw_aB$ iff $M \models Kw_aB$ iff $H, M \models Kw_aB$ (all what is used here, is Definition 3.9). For $\beta$ being a conjunction $\beta_1 \wedge \beta_2$, or a negation $\neg\beta_1$, (3.2) immediately follows from the inductive hypothesis that the theorem is proven for $\beta_1$ and $\beta_2$.

Moving on to $\mathcal{L}^{\beta}_{Kw}$, we have dealt with $\beta$, and conjunction and disjunction are immediate. So suppose the theorem proven for $\varphi_1 \in \mathcal{L}^{\beta}_{Kw}$, and consider $K_a\varphi_1$. If $G, M \models K_a\varphi_1$, by definition, we have that for all $N \in \mathcal{S}$ for which $M \approx_a N$, $G, N \models \varphi_1$. Let $N' \in \mathcal{S}'$ with $M \approx'_a N'$. We then know from the above that $M \approx_a N'$, and since the inductive hypothesis gives us $H, N' \models \varphi_1$, we conclude $H, M \models K_a\varphi_1$. □

*Proof of Proposition 3.20(1).* Let $G = (\mathcal{S}, \approx)$.

$$G \models [ab^{\mu}]Kw_cD \leftrightarrow [ab^{\mu'}]Kw_cD \iff \text{for every } M' \in \mathcal{S}: \\ G, M' \models [ab^{\mu}]Kw_cD \leftrightarrow [ab^{\mu'}]Kw_cD$$

Let $M = (T, \sim^M, V)$ be an arbitrary gossip situation in $\mathcal{S}$. Then:

$$\begin{aligned} G, M \models [ab^{\mu}]Kw_cD \leftrightarrow [ab^{\mu'}]Kw_cD &\iff G, M \models [ab^{\mu}]Kw_cD \text{ iff } G, M \models [ab^{\mu'}]Kw_cD \\ &\qquad \text{(from the semantics of '}\leftrightarrow\text{')} \\ &\iff G^{\mathsf{call}\mu}, M^{ab} \models Kw_cD \text{ iff } G^{\mathsf{call}\mu'}, M^{ab} \models Kw_cD \\ &\qquad \text{(from Definition 3.18)} \end{aligned}$$

for every $(G^{\mathsf{call}\mu}, M^{ab})$ and $(G^{\mathsf{call}\mu'}, M^{ab})$ such that $((G, M), (G^{\mathsf{call}\mu}, M^{ab})) \in [\![ab^{\mu}]\!]$ and $((G, M), (G^{\mathsf{call}\mu'}, M^{ab})) \in [\![ab^{\mu'}]\!]$, and $\mu, \mu' \in \{-, 0, +\}$, where, from Definition 3.18:

$G^{\mathsf{call}-} = (\mathcal{S}^-, \approx^-)$ and $\mathcal{S}^- = \{N^{ab} \mid N \in \mathcal{S}\}$;
$G^{\mathsf{call}0} = (\mathcal{S}^0, \approx^0)$ and $\mathcal{S}^0 = \{N^{cd} \mid N \in \mathcal{S} \text{ and } c \neq d \in \mathsf{Ag}\}$;
$G^{\mathsf{call}+} = (\mathcal{S}^+, \approx^+)$ and $\mathcal{S}^+ = \{N^{cd} \mid N \in \mathcal{S} \text{ and } c \neq d \in \mathsf{Ag}\} \cup \mathcal{S}$;
and $N^{ab} = (T, \sim^{N^{ab}}, V)$, and $N^{cd} = (T, \sim^{N^{cd}}, V)$.

Now, $G^{\mathsf{call}\mu}, M^{ab} \models Kw_cD$ iff $\exists \mathsf{Q} : \sim_c^{M^{ab}} = \equiv_{\mathsf{Q}}^{M^{ab}}$ and $D \in \mathsf{Q}$ (see Definition 3.6). But $M^{ab} \in \mathcal{S}^- \cap \mathcal{S}^0 \cap \mathcal{S}^+$. So $G^{call\mu}, M^{ab} \models Kw_cD$ iff $G^{call\mu'}, M^{ab} \models Kw_cD$, where $\mu \in \{-, 0, +\}$. And we therefore conclude that:

$$G, M \models [ab^{\mu}]Kw_cD \text{ iff } G, M \models [ab^{\mu'}]Kw_cD \iff G \models [ab^{\mu}]Kw_cD \leftrightarrow [ab^{\mu'}]Kw_cD$$

where $\mu \in \{-, 0, +\}$. □

*Proof of Proposition 3.20(2).* Let $G = (\mathcal{S}, \approx)$.

$$G \models [ab^0]\varphi \rightarrow [ab^-]\varphi \iff \text{for every } M' \in \mathcal{S} : G, M' \models [ab^0]\varphi \rightarrow [ab^-]\varphi$$

Let $M = (T, \sim^M, V)$ be an arbitrary gossip situation in $\mathcal{S}$. Then:

$$\begin{aligned} G, M \models [ab^0]\varphi \rightarrow [ab^-]\varphi &\iff \text{if } G, M \models [ab^0]\varphi \text{ then } G, M \models [ab^-]\varphi \\ &\qquad \text{(from the semantics of '}\rightarrow\text{')} \\ &\iff \text{if } G^{\mathsf{call}0}, M^{ab} \models \varphi \text{ then } G^{\mathsf{call}-}, M^{ab} \models \varphi \\ &\qquad \text{(from Definition 3.18)} \end{aligned} \qquad (\ddagger)$$

for every $(G^{\mathsf{call}0}, M^{ab})$ and $(G^{\mathsf{call}-}, M^{ab})$ such that $((G, M), (G^{\mathsf{call}0}, M^{ab})) \in [\![ab^0]\!]$ and $((G, M), (G^{\mathsf{call}-}, M^{ab})) \in [\![ab^-]\!]$, where, from Definition 3.18:

$G^{\mathsf{call}-} = (\mathcal{S}^-, \approx^-)$ and $\mathcal{S}^- = \{N^{ab} \mid N \in \mathcal{S}\}$;
$G^{\mathsf{call}0} = (\mathcal{S}^0, \approx^0)$ and $\mathcal{S}^0 = \{N^{cd} \mid N \in \mathcal{S} \text{ and } c \neq d \in \mathsf{Ag}\}$;

$$\text{and } N^{ab} = (T, \sim^{N^{ab}}, V), \text{ and } N^{cd} = (T, \sim^{N^{cd}}, V).$$

Now, notice that $M^{ab} \in \mathcal{S}^- \subseteq \mathcal{S}^0$. Therefore, from Theorem 3.21 we see that (‡) holds. And this leads to the conclusion that:

$$G \models [ab^0]\varphi \to [ab^-]\varphi \qquad \square$$

*Proof of Proposition 3.20(3).* Let $G = (\mathcal{S}, \approx)$.

$$G \models [ab^+]\varphi \to [ab^-]\varphi \iff \text{for every } M' \in \mathcal{S} : G, M' \models [ab^+]\varphi \to [ab^-]\varphi$$

Let $M = (T, \sim^M, V)$ be an arbitrary gossip situation in $\mathcal{S}$. Then:

$$\begin{aligned} G, M \models [ab^+]\varphi \to [ab^-]\varphi &\iff \text{if } G, M \models [ab^+]\varphi \text{ then } G, M \models [ab^-]\varphi \\ &\qquad\qquad \text{(from the semantics of '}\to\text{')} \\ &\iff \text{if } G^{\mathsf{call}+}, M^{ab} \models \varphi \text{ then } G^{\mathsf{call}-}, M^{ab} \models \varphi \\ &\qquad\qquad \text{(from Definition 3.18)} \end{aligned} \tag{‡}$$

for every $(G^{\mathsf{call}+}, M^{ab})$ and $(G^{\mathsf{call}-}, M^{ab})$ such that $((G, M), (G^{\mathsf{call}+}, M^{ab})) \in [\![ab^+]\!]$ and $((G, M), (G^{\mathsf{call}-}, M^{ab})) \in [\![ab^-]\!]$, where, from Definition 3.18:

$$\begin{aligned} &G^{\mathsf{call}-} = (\mathcal{S}^-, \approx^-) \text{ and } \mathcal{S}^- = \{N^{ab} \mid N \in \mathcal{S}\}; \\ &G^{\mathsf{call}+} = (\mathcal{S}^+, \approx^+) \text{ and } \mathcal{S}^+ = \{N^{cd} \mid N \in \mathcal{S} \text{ and } c \neq d \in \mathsf{Ag}\} \cup \mathcal{S}; \\ &\text{and } N^{ab} = (T, \sim^{N^{ab}}, V), \text{ and } N^{cd} = (T, \sim^{N^{cd}}, V). \end{aligned}$$

Now, notice that $M^{ab} \in \mathcal{S}^- \subseteq \mathcal{S}^+$. Therefore, from Theorem 3.21 we see that (‡) holds. And this leads to the conclusion that:

$$G \models [ab^+]\varphi \to [ab^-]\varphi \qquad \square$$

*Proof of Proposition 3.20(4).* Let $G = (\mathcal{S}, \approx)$ and let $|\mathsf{Ag}| = 3$. Then:

$$G \models [ab^-]\varphi \leftrightarrow [ab^0]\varphi \iff \text{for every } M' \in \mathcal{S} : G, M' \models [ab^-]\varphi \leftrightarrow [ab^0]\varphi$$

Let $M = (T, \sim^M, V)$ be an arbitrary gossip situation in $\mathcal{S}$. We show that:

$$G, M \models [ab^-]\varphi \leftrightarrow [ab^0]\varphi \quad (\dagger)$$

We begin with the **'$\Longrightarrow$'-direction** of (†), as follows.

$$\begin{aligned} G, M \models [ab^-]\varphi \to [ab^0]\varphi &\iff \text{if } G, M \models [ab^-]\varphi \text{ then } G, M \models [ab^0]\varphi \\ &\qquad\qquad \text{(from the semantics of '}\to\text{')} \\ &\iff \text{if } G^{\mathsf{call}-}, M^{ab} \models \varphi \text{ then } G^{\mathsf{call}0}, M^{ab} \models \varphi \\ &\qquad\qquad \text{(from Definition 3.18)} \end{aligned} \tag{‡}$$

for every $(G^{\mathsf{call}-}, M^{ab})$ and $(G^{\mathsf{call}0}, M^{ab})$ such that $((G,M),(G^{\mathsf{call}-},M^{ab})) \in [\![ab^-]\!]$ and $((G,M),(G^{\mathsf{call}0},M^{ab})) \in [\![ab^0]\!]$, where, from Definition 3.18:

$G^{\mathsf{call}-} = (\mathcal{S}^-, \approx^-)$ and $\mathcal{S}^- = \{N^{ab} \mid N \in \mathcal{S}\}$;
$G^{\mathsf{call}0} = (\mathcal{S}^0, \approx^0)$ and $\mathcal{S}^0 = \{N^{cd} \mid N \in \mathcal{S} \text{ and } c \neq d \in \mathsf{Ag}\}$;
and $N^{ab} = (T, \sim^{N^{ab}}, V)$, and $N^{cd} = (T, \sim^{N^{cd}}, V)$.

We now proceed by induction on $\varphi \in \mathcal{L}^{\beta}_{Kw}$.

**Base Cases on $\mathcal{L}^{\beta}_{Kw}$. Case $\varphi$ is $\beta \in \mathcal{L}_\beta$.**

We proceed by induction on $\beta$, as follows.

**Base Case on $\mathcal{L}_\beta$.**

We consider two base cases: $\beta$ is $Kw_aB$ and $\beta$ is $\neg Kw_aB$.

Suppose $\beta$ is $Kw_aB$. Then from (‡), this is the same as saying that if $G^{\mathsf{call}-}, M^{ab} \models Kw_aB$ then $G^{\mathsf{call}0}, M^{ab} \models Kw_aB$, for $a, b \in \mathsf{Ag}$. The proof follows from Proposition 3.20(1) which says that:

$$G^{\mathsf{call}\mu}, M^{ab} \models Kw_aB \text{ iff } G^{\mathsf{call}\mu'}, M^{ab} \models Kw_aB \quad \text{for } a, b \in \mathsf{Ag} \text{ and } \mu, \mu' \in \{-, 0, +\}$$

Therefore $G^{\mathsf{call}-}, M^{ab} \models Kw_aB$ iff $G^{\mathsf{call}0}, M^{ab} \models Kw_aB$.

For the second base case, suppose that $\beta$ is $\neg Kw_aB$. Again from Proposition 3.20(1), we see that since $G^{\mathsf{call}-}, M^{ab} \models Kw_aB$ iff $G^{\mathsf{call}0}, M^{ab} \models Kw_aB$. Therefore, by contraposition, $G^{\mathsf{call}-}, M^{ab} \not\models Kw_aB$ *implies* $G^{\mathsf{call}0}, M^{ab} \not\models Kw_aB$, which is equivalent to saying that:

$$G^{\mathsf{call}-}, M^{ab} \models \neg Kw_aB \textit{ implies } G^{\mathsf{call}0}, M^{ab} \models \neg Kw_aB \quad \text{(from the semantics of the '}\neg\text{' operator)}$$

**Inductive Hypothesis on $\mathcal{L}_\beta$** For every $\beta', \beta'' \in \mathcal{L}_\beta \subset \mathcal{L}^*_{Kw}$, it is the case that:

1. if $G^{\mathsf{call}-}, M^{ab} \models \beta'$ then $G^{\mathsf{call}0}, M^{ab} \models \beta'$
2. if $G^{\mathsf{call}-}, M^{ab} \models \neg\beta'$ then $G^{\mathsf{call}0}, M^{ab} \models \neg\beta'$
3. if $G^{\mathsf{call}-}, M^{ab} \models \beta''$ then $G^{\mathsf{call}0}, M^{ab} \models \beta''$
4. if $G^{\mathsf{call}-}, M^{ab} \models \neg\beta''$ then $G^{\mathsf{call}0}, M^{ab} \models \neg\beta''$

**Inductive Step on $\mathcal{L}_\beta$.** We distinguish the following cases:

**Case $\beta$ is $\neg\beta'$.** This case is straightforward from the inductive hypothesis.

**Case $\beta$ is $(\beta' \wedge \beta'')$.** By the inductive hypothesis we have that:

$$\text{if } G^{\mathsf{call}-}, M^{ab} \models \beta' \text{ then } G^{\mathsf{call}0}, M^{ab} \models \beta'$$
$$\text{and if } G^{\mathsf{call}-}, M^{ab} \models \beta'' \text{ then } G^{\mathsf{call}0}, M^{ab} \models \beta''$$

And that is equivalent to saying that:

$$\text{if } G^{\mathsf{call}-}, M^{ab} \models (\beta' \wedge \beta'') \text{ then } G^{\mathsf{call}0}, M^{ab} \models (\beta' \wedge \beta'')$$
(by propositional logic).

**Inductive Hypothesis on $\mathcal{L}^{\beta}_{Kw}$** For every $\varphi', \varphi'' \in \mathcal{L}^{\beta}_{Kw}$, it is the case that:

1. if $G^{\mathsf{call}-}, M^{ab} \models \varphi'$ then $G^{\mathsf{call}0}, M^{ab} \models \varphi'$
2. if $G^{\mathsf{call}-}, M^{ab} \models \varphi''$ then $G^{\mathsf{call}0}, M^{ab} \models \varphi''$

**Inductive Step on $\mathcal{L}^{\beta}_{Kw}$.** We distinguish the following cases:

**Case $\varphi$ is $(\varphi' \vee \varphi'')$.** By propositional logic, the inductive hypothesis implies that the following holds:

$$\text{if } G^{\mathsf{call}-}, M^{ab} \models \varphi' \text{ then } G^{\mathsf{call}0}, M^{ab} \models \varphi'$$
$$\text{or if } G^{\mathsf{call}-}, M^{ab} \models \varphi'' \text{ then } G^{\mathsf{call}0}, M^{ab} \models \varphi''$$

And that is equivalent to saying that:

$$\text{if } G^{\mathsf{call}-}, M^{ab} \models (\varphi' \vee \varphi'') \text{ then } G^{\mathsf{call}0}, M^{ab} \models (\varphi' \vee \varphi'')$$
(by propositional logic).

**Case $\varphi$ is $(\varphi' \wedge \varphi'')$.** Straightforward from the inductive hypothesis:

$$\text{if } G^{\mathsf{call}-}, M^{ab} \models \varphi' \text{ then } G^{\mathsf{call}0}, M^{ab} \models \varphi'$$
$$\text{and if } G^{\mathsf{call}-}, M^{ab} \models \varphi'' \text{ then } G^{\mathsf{call}0}, M^{ab} \models \varphi''$$

And that is equivalent to saying that:

$$\text{if } G^{\mathsf{call}-}, M^{ab} \models (\varphi' \wedge \varphi'') \text{ then } G^{\mathsf{call}0}, M^{ab} \models (\varphi' \wedge \varphi'')$$
(by propositional logic).

**Case $\varphi$ is $K_c\varphi'$.** Suppose that $G^{\mathsf{call}-}, M^{ab} \models K_c\varphi'$. That is, for every $M'$, $M^{ab} \approx_c^- M'$ implies $G^{\mathsf{call}-}, M' \models \varphi'$. We now want to show that $G^{\mathsf{call}0}, M^{ab} \models K_c\varphi'$. That is, for every $M''$, $M^{ab} \approx_c^0 M''$ implies $G^{\mathsf{call}0}, M'' \models \varphi'$. Let $\mathsf{Set_1} = \{M'' \mid M'' \in$

$\mathcal{S}^0$ and $M^{ab} \approx_c^0 M''\}$. Let $\mathsf{Set_2} = \{M' \mid M' \in \mathcal{S}^-$ and $M^{ab} \approx_c^- M'\}$. We proceed by case distinctions, as follows.

**Case 1:** Consider the case where agent $c$ was not involved in the actual call $ab$, and recall that $|\mathsf{Ag}| = 3$. Then in both $\mathcal{S}^-$ and $\mathcal{S}^0$ agent $c$ cannot distinguish between those gossip situations in which it was not involved in the call, namely, $M^{ab}$ and $M^{ba}$ (in those two gossip situations, the set of secrets known by agent $c$ remained the same). Therefore $\mathsf{Set_1} = \{M^{ab}, M^{ba}\} = \mathsf{Set_2}$.

**Case 2:** On the other hand, suppose agent $c$ was involved in the actual call. Without loss of generality, let the actual call be $ac$. Then in both $\mathcal{S}^-$ and $\mathcal{S}^0$ agent $c$ cannot distinguish between those gossip situations in which it called with the same other agent, namely, $M^{ac}$ and $M^{ca}$ (in those two gossip situations, the set of secrets known by agent $c$ are the same after the call). Therefore $\mathsf{Set_1} = \{M^{ac}, M^{ca}\} = \mathsf{Set_2}$.

From $\mathsf{Set_1} = \mathsf{Set_2}$ in both Case 1 and Case 2, and from the inductive hypothesis, we see that, if for every $M'$ such that $M^{ab} \approx_c^- M'$, $G^{\mathsf{call}-}, M' \models \varphi'$, then also for every $M''$ such that $M^{ab} \approx_c^0 M''$, $G^{\mathsf{call}0}, M'' \models \varphi'$. And this is equivalent to saying that if $G^{\mathsf{call}-}, M^{ab} \models K_c\varphi'$ then $G^{\mathsf{call}0}, M^{ab} \models K_c\varphi'$, from the semantics of '$K_a$'.

This concludes the inductive argument.

Now we consider the **'$\Longleftarrow$'-direction** of (†), as follows.

$$G \models [ab^0]\varphi \to [ab^-]\varphi \text{ if } |\mathsf{Ag}| = 3$$

The proof of this case follows from Proposition 3.20(2).

We therefore conclude that it is the case that:

$$G \models [ab^-]\varphi \leftrightarrow [ab^0]\varphi \text{ if } |\mathsf{Ag}| = 3$$

This now concludes the proof of Proposition 3.20(4). □

Proposition 3.22 (below) highlights some specific properties of each of the basic call modes.

**Proposition 3.22.** *Let a gossip model $G = (\mathcal{S}, \approx)$ for at least four agents $a, b, c, d$ be given, and let the mode $\mu$ of the call be a variable over $\{0, +, -\}$. Then:*

1. $G \models K_cKw_bD \to K_c[ab^\mu]Kw_aD$ *for* $\mu \in \{0, +, -\}$;
2. $G \models [ab^\mu]K_c\neg init$ *only for* $\mu \in \{0, -\}$;
3. $G \models \neg K_cKw_aB \to [ab^\mu]\neg K_cKw_aB$ *only for* $\mu \in \{+\}$;
4. $G \models K_cKw_bD \to [ab^\mu]K_cKw_aD$ *only for* $\mu \in \{-\}$;

5. $G \models init \to [ab^{\mu}]\neg K_c \neg init$ *only for* $\mu \in \{+\}$, *and* $c \notin \{a, b\}$.

6. $G \models init \to [ab^{\mu}](K_c \bigvee_{x \neq y} Kw_x Y \wedge \neg \bigvee_{x \neq y} K_c Kw_x Y)$ *only for* $\mu \in \{0\}$;

*where init is a designated atom denoting the initial situation.*

Items 1 - 6 show that indeed, all modes of making calls are different. Loosely speaking, the first item says that any agent ($c$) knows that any call between $a$ and $b$ brings about that both get to know each other's secrets. Contrast this to item 4: only for the mode $\mu = -$, agent $c$ knows that the call happens and remembers its predicted effects. Item 2, which is only true for the modes 0 and $-$, says that a call in the initial situation causes some agent in the scenario to learn some new secret. Note that the fact that at least one agent learns at least one secret given an arbitrary state is not generally true: it may be that a call takes place between two agents who are unable to tell each other anything new. Considering Item 3, agent $c$ can only know that $a$ learned a new secret as the consequence of the call when the call is made in $-$-mode, or if $c$ is involved in the call. Item 5 says that after an $ab^{+}$-call, an outsider does not know anything has happened. Finally, item 6 tells us that after an $ab^{0}$-call in the initial situation, an outsider knows that somebody learned something, but the outsider does not know who learned something. We now present the proof of proposition 3.22.

*Proof of Proposition 3.22(1).* Let $G = (\mathcal{S}, \approx)$, and let $\mu \in \{0, +, -\}$. Then:

$$G \models K_c Kw_b D \to K_c[ab^{\mu}]Kw_a D \iff \text{for every } M' \in \mathcal{S}: \\ G, M' \models K_c Kw_b D \to K_c[ab^{\mu}]Kw_a D$$

Let $M = (T, \sim^M, V)$ be an arbitrary gossip situation in $\mathcal{S}$. We show that:

$$G, M \models K_c Kw_b D \to K_c[ab^{\mu}]Kw_a D$$

Now:

$$G, M \models K_c Kw_b D \to K_c[ab^{\mu}]Kw_a D \iff \\ \text{if } G, M \models K_c Kw_b D \text{ then } G, M \models K_c[ab^{\mu}]Kw_a D \quad \text{(from the semantics of '}\to\text{')}$$

Suppose $G, M \models K_c Kw_b D$ holds. From the semantics of '$K_c$' operator, this is equivalent to saying that, for every $M'$ such that $M \approx_c M'$, $\quad G, M' \models Kw_b D$. That is, for all such $M'$, $\quad \sim_b^{M'} = \equiv_{\mathsf{Q}_b}^{M'}$ and $D \in \mathsf{Q}_b \qquad (\dagger)$

We want to show that $G, M \models K_c[ab^{\mu}]Kw_a D$. That is, for every $M'$ such that $M \approx_c M'$, $G, M' \models [ab^{\mu}]Kw_a D$. This is equivalent to saying that $G^{\mathsf{call}\mu}, M'^{ab} \models Kw_a D$ for every $(G^{\mathsf{call}\mu}, M'^{ab})$ such that $((G, M'), (G^{\mathsf{call}\mu}, M'^{ab})) \in [\![ab^{\mu}]\!]$, where, from Definition 3.18:

$G^{\mathsf{call}-} = (\mathcal{S}^-, \approx^-)$ and $\mathcal{S}^- = \{N^{ab} \mid N \in \mathcal{S}\}$;
$G^{\mathsf{call}0} = (\mathcal{S}^0, \approx^0)$ and $\mathcal{S}^0 = \{N^{cd} \mid N \in \mathcal{S} \text{ and } c \neq d \in \mathsf{Ag}\}$;
$G^{\mathsf{call}+} = (\mathcal{S}^+, \approx^+)$ and $\mathcal{S}^+ = \{N^{cd} \mid N \in \mathcal{S} \text{ and } c \neq d \in \mathsf{Ag}\} \cup \mathcal{S}$;
and $N^{ab} = (T, \sim^{N^{ab}}, V)$, and $N^{cd} = (T, \sim^{N^{cd}}, V)$.

Now, for such $M'$, $G, M' \models Kw_bD$ (from (†)). Furthermore, for all such $M'^{ab}$, $M'^{ab} \in \mathcal{S}^-$ (since $ab$ is the actual call), and $M'^{ab} \in \mathcal{S}^0$, and $M'^{ab} \in \mathcal{S}^+$ (see Definition 3.18). Also, for all such $M'^{ab}$: $\sim_a^{M'^{ab}} = \equiv_{\mathsf{Q}_{ab}}^{M'^{ab}} = \sim_a^{M'} \cap \sim_b^{M'}$ (see Definition 3.16), and from Lemma 3.17, $D \in \mathsf{Q}_{ab}$ since $D \in \mathsf{Q}_b$ (from (†)). Therefore, from Theorem 3.8, we conclude that for all such $M'^{ab}$:

$$\begin{aligned} G^{\mathsf{call}\mu}, M'^{ab} \models Kw_aD &\iff G, M' \models [ab^\mu]Kw_aD, \text{ for every } M' \text{ such that } M \approx_c M' \\ &\iff G, M \models K_c[ab^\mu]Kw_aD \quad \text{(semantics of '}K_c\text{' operator)} \\ &\iff G \models K_cKw_bD \to K_c[ab^\mu]Kw_aD, \text{ where } \mu \in \{0, +, -\}. \end{aligned}$$

□

*Proof of Proposition 3.22(2).* Let $G = (\mathcal{S}, \approx)$, and let $M_1 = (T, \sim^{M_1}, V)$ be the initial gossip situation.

$$G \models [ab^\mu]K_c\neg init, \text{ only for } \mu \in \{0, -\} \iff \text{for every } M' \in \mathcal{S}: \\ G, M' \models [ab^\mu]K_c\neg init, \text{ only for } \mu \in \{0, -\}$$

Let $M = (T, \sim^M, V)$ be an arbitrary gossip situation in $\mathcal{S}$. We show that:

$$G, M \models [ab^\mu]K_c\neg init, \text{ only for } \mu \in \{0, -\}$$

Now, only for $\mu \in \{0, -\}$:

$$\begin{aligned} G, M \models [ab^\mu]K_c\neg init &\iff G^{\mathsf{call}\mu}, M^{ab} \models K_c\neg init \\ &\iff G^{\mathsf{call}-}, M^{ab} \models K_c\neg init \\ &\text{and} \quad G^{\mathsf{call}0}, M^{ab} \models K_c\neg init \\ &\text{and} \quad G^{\mathsf{call}+} \not\models K_c\neg init \end{aligned} \tag{‡}$$

for every $(G^{\mathsf{call}-}, M^{ab})$ and $(G^{\mathsf{call}0}, M^{ab})$ such that $((G, M), (G^{\mathsf{call}-}, M^{ab})) \in [\![ab^-]\!]$ and $((G, M), (G^{\mathsf{call}0}, M^{ab})) \in [\![ab^0]\!]$, where, from Definition 3.18:

$G^{\mathsf{call}-} = (\mathcal{S}^-, \approx^-)$ and $\mathcal{S}^- = \{N^{ab} \mid N \in \mathcal{S}\}$;
$G^{\mathsf{call}0} = (\mathcal{S}^0, \approx^0)$ and $\mathcal{S}^0 = \{N^{cd} \mid N \in \mathcal{S} \text{ and } c \neq d \in \mathsf{Ag}\}$;
$G^{\mathsf{call}+} = (\mathcal{S}^+, \approx^+)$ and $\mathcal{S}^+ = \{N^{cd} \mid N \in \mathcal{S} \text{ and } c \neq d \in \mathsf{Ag}\} \cup \mathcal{S}$;
and $N^{ab} = (T, \sim^{N^{ab}}, V)$, and $N^{cd} = (T, \sim^{N^{cd}}, V)$

Recall that the initial gossip situation $M_1$ is such that: $\sim_a^{M_1} = \equiv_{\mathsf{Q}_a}^{M_1}$, where $\mathsf{Q}_a = \{A\}$, for all $a \in \mathsf{Ag}$ and where $A$ is the unique secret of agent $a$.

From (‡), let us consider whether $G^{\mathsf{call}-}, M^{ab} \models K_c \neg init$ holds. From the semantics of $ab^-$ call in a gossip model (see Definition 3.18), for every $N^{ab} \in \mathcal{S}^-$, $\sim_a^{N^{ab}} = \equiv_{\mathsf{Q}_{ab}}^{N^{ab}}$. But $\sim_a^{N^{ab}} = \sim_a^N \cap \sim_b^N$, and from Lemma 3.17, $\mathsf{Q}_{ab} = \mathsf{Q}_a \cup \mathsf{Q}_b$. Therefore after the $ab^-$ call, no agent considers it possible that there is a situation $N^{ab} \in \mathcal{S}^-$ and $\sim_a^{N^{ab}} = \equiv_{\mathsf{Q}_a}^{N^{ab}}$, where $\mathsf{Q}_a = \{A\}$, for all $a \in \mathsf{Ag}$. Therefore:

$$\begin{array}{rcl} G^{\mathsf{call}-}, M^{ab} \models K_c \neg init & \iff & G, M \models [ab^-] K_c \neg init \\ & \iff & G \models [ab^-] K_c \neg init \end{array}$$

From (‡), let us consider whether $G^{\mathsf{call}0}, M^{ab} \models K_c \neg init$ holds. Similar to the foregoing case of $ab^-$, from the semantics of $ab^0$ call in a gossip model (see Definition 3.18), for every $N^{cd} \in \mathcal{S}^0$, $\sim_c^{N^{cd}} = \equiv_{\mathsf{Q}_{cd}}^{N^{cd}}$. But $\sim_c^{N^{cd}} = \sim_c^N \cap \sim_d^N$, and from Lemma 3.17, $\mathsf{Q}_{cd} = \mathsf{Q}_c \cup \mathsf{Q}_d$. Therefore after the $ab^0$ call, no agent considers it possible that there is a situation $N^{cd} \in \mathcal{S}^0$ where $\sim_c^{N^{cd}} = \equiv_{\mathsf{Q}_c}^{N^{cd}}$ and $\mathsf{Q}_c = \{C\}$, for all $c \in \mathsf{Ag}$. Therefore:

$$\begin{array}{rcl} G^{\mathsf{call}0}, M^{ab} \models K_c \neg init & \iff & G, M \models [ab^0] K_c \neg init \\ & \iff & G \models [ab^0] K_c \neg init \end{array}$$

Finally, from (‡), to see that $G^{\mathsf{call}+}, M^{ab} \not\models K_c \neg init$ holds. From the semantics of $ab^+$ call in a gossip model (see Definition 3.18), $\mathcal{S}^+ = \mathcal{S}^0 \cup \mathcal{S}$. So the initial gossip situation $M_1$ is in $\mathcal{S}^+$. Now, consider a gossip situation $N^{ab} \in \mathcal{S}^+$ such that $c \notin \{a, b\} \subset \mathsf{Ag}$, we have that $M_1 \approx_c^+ N^{ab}$ (agent $c$ was not involved in the $ab$ call, so in the $+$-mode agent $c$ considers it possible that no call took place and as such the resulting situation could still be $M_1$). Therefore:

$$\begin{array}{rcl} G^{\mathsf{call}+}, M^{ab} \not\models K_c \neg init & \iff & G, M \not\models [ab^+] K_c \neg init \\ & \iff & G \not\models [ab^+] K_c \neg init \end{array}$$

□

*Proof of Proposition 3.22(3).* Let $G = (\mathcal{S}, \approx)$. We want to show that:

$$G \models \neg K_c K w_a B \to [ab^\mu] \neg K_c K w_a B, \text{ only for } \mu \in \{+\} \qquad (‡)$$

That is,

1. $G \models \neg K_c K w_a B \to [ab^+] \neg K_c K w_a B$
2. $G \not\models \neg K_c K w_a B \to [ab^0] \neg K_c K w_a B$
3. $G \not\models \neg K_c K w_a B \to [ab^-] \neg K_c K w_a B$

For Item 1,

$$G \models \neg K_c K w_a B \to [ab^+]\neg K_c K w_a B \iff \text{for every } M' \in \mathcal{S}: \\ G, M' \models \neg K_c K w_a B \to [ab^+]\neg K_c K w_a B$$

Let $M = (T, \sim^M, V)$ be an arbitrary gossip situation in $\mathcal{S}$. We show that:

$$G, M \models \neg K_c K w_a B \to [ab^+]\neg K_c K w_a B \iff \\ \text{if } G, M \models \neg K_c K w_a B \text{ then } G, M \models [ab^+]\neg K_c K w_a B \\ \text{(from the semantics of '}\to\text{')}$$

Suppose $G, M \models \neg K_c K w_a B$ holds. That is, there is some $M' \in \mathcal{S}$ such that $M \approx_c M'$, and $G, M' \models \neg K w_a B$. That is, for such $M'$, $\sim_a^{M'} = \equiv_{\mathsf{Q}_a}^{M'}$ and $B \notin \mathsf{Q}_a$ (†)

Now we want to show that: $G, M \models [ab^+]\neg K_c K w_a B$, that is, $G^{\mathsf{call}+}, M^{ab} \models \neg K_c K w_a B$, for every $(G^{\mathsf{call}+}, M^{ab})$ such that $((G, M), (G^{\mathsf{call}+}, M^{ab})) \in [\![ab^+]\!]$ where, from Definition 3.18:

$$G^{\mathsf{call}+} = (\mathcal{S}^+, \approx^+) \text{ and } \mathcal{S}^+ = \{N^{cd} \mid N \in \mathcal{S} \text{ and } c \neq d \in \mathsf{Ag}\} \cup \mathcal{S}; \\ \text{and } N^{cd} = (T, \sim^{N^{cd}}, V)$$

Now, $G^{\mathsf{call}+}, M^{ab} \models \neg K_c K w_a B$ holds if and only if there is some $N \in \mathcal{S}^+$ such that $M^{ab} \approx_c N$ and $G^{\mathsf{call}+}, N \models \neg K w_a B$. Now, to see that $G^{\mathsf{call}+}, M^{ab} \models \neg K_c K w_a B$ holds, recall (from (†)) that $M, M' \in \mathcal{S}$ and $M \approx_c M'$. But also $M, M' \in \mathcal{S}^+$ (due to the possible skip action at $M$ and $M'$, these situations are also possible situations in the domain of $G^{\mathsf{call}+}$ (see Definition 3.18)). Since $c \notin \{a, b\}$, that is, agent $c$ was not in the $ab$ call at $G, M$, we see that $M^{ab} \approx_c M \approx_c M'$. That is, $\sim_c^M = \sim_c^{M'} = \sim_c^{M^{ab}}$ (see Definition 3.16 and Definition 3.18). But from (†), $\sim_a^{M'} = \equiv_{\mathsf{Q}_a}^{M'}$ and $B \notin \mathsf{Q}_a$. So we see that $M^{ab} \approx_c M'$, that is, agent $c$ considers $M'$ possible at $(G^{\mathsf{call}+}, M^{ab})$, and $G^{\mathsf{call}+}, M' \models \neg K w_a B$. And so we conclude that:

$$G^{\mathsf{call}+}, M^{ab} \models \neg K_c K w_a B \iff G, M \models [ab^+]\neg K_c K w_a B$$

For Item 2. Let $G = (\mathcal{S}, \approx)$ be such that for every $N \in \mathcal{S}$, $G, N \models K w_x B$ holds only for every $x \neq a \in \mathsf{Ag}$. Note that $G, N \models \neg K_c K w_a B$ holds since there is no gossip situation in $\mathcal{S}$ where $K w_a B$ holds. To see that $G^{\mathsf{call}0}, M^{ab} \not\models \neg K_c K w_a B$ holds, consider a call $ab$ involving agent $a$ at one such $N = M$ in $G$. Let the resulting gossip model after this call be $G^{\mathsf{call}0} = (\mathcal{S}^0, \approx^0)$. Now, this $G^{\mathsf{call}0}$ is such that for every $N^{cd} \in \mathcal{S}^0$, $G^{\mathsf{call}0}, N^{cd} \models K w_x B$ for every $x \in \mathsf{Ag}$. This is the case since for every such $N^{cd}$, there is some $\mathsf{Q}$ such that $\sim_x^{N^{cd}} = \equiv_{\mathsf{Q}}^{N^{cd}}$ and $B \in \mathsf{Q}$, where $x, c, d \in \mathsf{Ag}$ (that is, no matter the calling pair, the resulting situation is that in which every agent knows the secret of agent $B$). But $M^{ab} \in \mathcal{S}^0$, and since agent $c$ does not consider it possible that there is any

gossip situation in $\mathcal{S}^0$ in which agent $a$ does not know the secret of agent $b$, we therefore obtain $G^{\mathsf{call}0}, M^{ab} \not\models \neg K_c K w_a B \iff G, M \not\models [ab^0]\neg K_c K w_a B$. And from the foregoing we conclude that Item 2 holds.

Finally, for Item 3. Consider any $G = (\mathcal{S}, \approx)$ and any $M \in \mathcal{S}$ such that $G, M \models \neg K_c K w_a B$. To see that $G^{\mathsf{call}-}, M^{ab} \not\models \neg K_c K w_a B$, consider that $G^{\mathsf{call}-}, M^{ab} \not\models \neg K_c K w_a B$ is equivalent to $G^{\mathsf{call}-}, M^{ab} \models K_c K w_a B$. Now, from the semantics of $ab^-$ call in a gossip model (see Definition 3.18), for every $N^{ab} \in \mathcal{S}^-$, $\sim_a^{N^{ab}} = \equiv_{\mathsf{Q}_{ab}}^{N^{ab}}$ and $B \in \mathsf{Q}_{ab}$. This is because $B \in \mathsf{Q}_b$ and $\sim_a^{N^{ab}} = \sim_a^N \cap \sim_b^N$, and from Lemma 3.17, $\mathsf{Q}_{ab} = \mathsf{Q}_a \cup \mathsf{Q}_b$. Therefore agent $c$ does not consider it possible that there is some gossip situation $M^{ab}$ after the $ab^-$ call such that $G^{\mathsf{call}-}, M^{ab} \models \neg K w_a B$. So we conclude that $G^{\mathsf{call}-}, M^{ab} \not\models \neg K_c K w_a B$, which concludes the proof of Item 3.

Therefore, from the proof of Item 1, 2 and 3 we conclude that (‡) holds. □

*Proof of Proposition 3.22(4).* Let $G = (\mathcal{S}, \approx)$, then only for $\mu \in \{-\}$:

$$\begin{array}{lll} G \models K_c K w_b D \to [ab^\mu] K_c K w_a D & \iff & G \models K_c K w_b D \to [ab^-] K_c K w_a D \\ & & \text{and} \quad G \not\models K_c K w_b D \to [ab^0] K_c K w_a D \qquad (\ddagger) \\ & & \text{and} \quad G \not\models K_c K w_b D \to [ab^+] K_c K w_a D \end{array}$$

From (‡), we begin by showing that:

$$G \models K_c K w_b D \to [ab^-] K_c K w_a D$$

Now:

$$G \models K_c K w_b D \to [ab^-] K_c K w_a D \iff \text{for every } M' \in \mathcal{S}: \\ G, M' \models K_c K w_b D \to [ab^-] K_c K w_a D$$

Let $M = (T, \sim^M, V)$ be an arbitrary gossip situation in $\mathcal{S}$. We show that:

$$G, M \models K_c K w_b D \to [ab^-] K_c K w_a D \iff \\ \text{if } G, M \models K_c K w_b D \text{ then } G, M \models [ab^-] K_c K w_a D \quad (\text{from the semantics of } `\to')$$

Suppose $G, M \models K_c K w_b D$ holds. That is, for every $M' \in \mathcal{S}$ such that $M \approx_c M'$, $G, M' \models K w_b D$ if and only if for all such $M'$, $\sim_b^{M'} = \equiv_{\mathsf{Q}_b}^{M'}$ and $D \in \mathsf{Q}_b$ (†)

Now we want to show that: $G, M \models [ab^-] K_c K w_a D$. From Definition 3.18,

$$G, M \models [ab^-] K_c K w_a D \iff G^{\mathsf{call}-}, M^{ab} \models K_c K w_a D \\ \text{for every } (G^{\mathsf{call}-}, M^{ab}) \text{ such that } ((G, M), (G^{\mathsf{call}-}, M^{ab})) \in [\![ab^-]\!] \text{ where,}$$

$$G^{\mathsf{call}-} = (\mathcal{S}^-, \approx^-) \text{ and } \mathcal{S}^- = \{N^{ab} \mid N \in \mathcal{S}\};$$
$$\text{and } N^{ab} = (T, \sim^{N^{ab}}, V)$$

Now, from the semantics of $ab^-$ call in a gossip model (see Definition 3.18), for every $N^{ab} \in \mathcal{S}^-$, $\sim_a^{N^{ab}} = \equiv_{\mathsf{Q}_{ab}}^{N^{ab}}$ and (from Lemma 3.17) $\mathsf{Q}_{ab} = \mathsf{Q}_a \cup \mathsf{Q}_b$. But from (†), $D \in \mathsf{Q}_b$, so $D \in \mathsf{Q}_{ab}$. Therefore agent $c$ does not consider it possible that there is a gossip situation $N^{ab} \in \mathcal{S}^-$ after the $ab^-$ call at $(G^{\mathsf{call}-}, M^{ab})$ such that $G^{\mathsf{call}-}, M^{ab} \models \neg Kw_a D$. So we conclude that:

$$G^{\mathsf{call}-}, M^{ab} \models K_c Kw_a D \tag{††}$$

Now consider (‡). We provide a counter example to show that $G \not\models K_c Kw_b D \rightarrow [ab^0]K_c Kw_a D$. Consider an $M, M' \in \mathcal{S}$ such that $M \approx_c M'$, $G, M \models Kw_b D$ and $G, M' \models \neg Kw_a D$. We want to show that $G^{\mathsf{call}0}, M^{ab} \not\models K_c Kw_a D$. That is, there is some $N \in \mathcal{S}^0$, $M^{ab} \approx_c^0 N$ and $G, N \models \neg Kw_a D$ (see Definition 3.18 for the definition of $\mathcal{S}^0$). After the $ab^0$ call at $(G, M)$, there are $N', N'' \in \mathcal{S}^0$ such that $N' = M^{ab}$ (that is, the gossip situation due to a call $ab^0$ at the gossip situation $M$) and $N'' = M'^{bd}$ (that is, the gossip situation due to a call $bd^0$ at the gossip situation $M'$). But $\sim_c^M = \sim_c^{M'} = \sim_c^{M^{ab}} = \sim_c^{M'^{bd}}$ (since $M \approx_c M'$ and the set of secrets known by agent $c$ is the same in $M^{ab}$ and $M'^{bd}$ as it is in $M$ and $M'$, see Definition 3.7 and Definition 3.16). Therefore $M^{ab} \approx_c^0 M'^{bd}$. Likewise, since $a \notin \{b, d\}$, $\sim_a^{M'^{bd}} = \equiv_{\mathsf{Q}'}^{M'^{bd}}$ and $D \notin \mathsf{Q}'$. We therefore conclude that there is some gossip situation, namely, $M'^{bd}$ such that $M^{ab} \approx_a^0 M'^{bd}$ and $G^{\mathsf{call}0}, M'^{bd} \models \neg Kw_a D$. And therefore, for such $M$ and $M'$, $G, M \not\models K_c Kw_b D \rightarrow [ab^0]K_c Kw_a D$, and therefore:

$$G \not\models K_c Kw_b D \rightarrow [ab^0]K_c Kw_a D \tag{†††}$$

Finally, consider (‡) again. We provide a counter example to show that $G \not\models K_c Kw_b D \rightarrow [ab^+]K_c Kw_a D$. Consider an $M, M' \in \mathcal{S}$ such that $M \approx_c M'$, $G, M \models Kw_b D$ and $G, M' \models \neg Kw_a D$. We want to show that $G^{\mathsf{call}+}, M^{ab} \not\models K_c Kw_a D$. That is, there is some $N \in \mathcal{S}^+$, $M^{ab} \approx_c^+ N$ and $G, N \models \neg Kw_a D$ (see Definition 3.18 for the definition of $\mathcal{S}^+$). After the $ab^+$ call at $(G, M)$, there are $N', N'' \in \mathcal{S}^+$ such that $N' = M^{ab}$ (that is, the gossip situation due to a call $ab^+$ at the gossip situation $M$) and $N'' = M'^{bd}$ (that is, the gossip situation due to a call $bd^+$ at the gossip situation $M'$). But $\sim_c^M = \sim_c^{M'} = \sim_c^{M^{ab}} = \sim_c^{M'^{bd}}$ (since $M \approx_c M'$ and the set of secrets known by agent $c$ is the same in $M^{ab}$ and $M'^{bd}$ as it is in $M$ and $M'$, see Definition 3.7 and Definition 3.16). Therefore $M^{ab} \approx_c^+ M'^{bd}$. Likewise, since $a \notin \{b, d\}$, $\sim_a^{M'^{bd}} = \equiv_{\mathsf{Q}'}^{M'^{bd}}$ and $D \notin \mathsf{Q}'$. We therefore conclude that there is some gossip situation, namely, $M'^{bd}$ such that $M^{ab} \approx_a^+ M'^{bd}$ and $G^{\mathsf{call}+}, M'^{bd} \models \neg Kw_a D$. And therefore, for such $M$ and $M'$, $G, M \not\models K_c Kw_b D \rightarrow [ab^+]K_c Kw_a D$, and therefore:

$$G \not\models K_c Kw_b D \rightarrow [ab^+]K_c Kw_a D \tag{††††}$$

Therefore, from (††),(†††) and (††††) we conclude that (‡) holds, and that:

$$G \models K_c K w_a D \to [ab^\mu] K_c K w_a D, \quad \text{only for } \mu \in \{-\} \qquad \square$$

*Proof of Proposition 3.22(5).* Let $G = (\mathcal{S}, \approx)$. Let the initial gossip situation be $M_1 = (T, \sim^{M_1}, V)$, where $M_1 \in \mathcal{S}$, then only for $\mu \in \{+\}$:

$$\begin{array}{rcll} G \models \mathit{init} \to [ab^\mu] \neg K_c \neg \mathit{init} & \iff & G \models \mathit{init} \to [ab^+] \neg K_c \neg \mathit{init} & \\ & & \text{and} \quad G \not\models \mathit{init} \to [ab^-] \neg K_c \neg \mathit{init} & (\ddagger) \\ & & \text{and} \quad G \not\models \mathit{init} \to [ab^0] \neg K_c \neg \mathit{init} & \end{array}$$

From (‡), we begin by showing that:

$$G \models \mathit{init} \to [ab^+] \neg K_c \neg \mathit{init}$$

Now:

$$\begin{array}{rcl} G \models \mathit{init} \to [ab^+] \neg K_c \neg \mathit{init} & \iff & G, M_1 \models [ab^+] \neg K_c \neg \mathit{init} \\ & & \text{(assuming the initial gossip situation)} \\ & \iff & G^{\mathsf{call}+}, M_1^{ab} \models \neg K_c \neg \mathit{init} \\ & & \text{(from Definition 3.18)} \end{array}$$

$G^{\mathsf{call}+}, M_1^{ab} \models \neg K_c \neg \mathit{init}$ if and only if there is some gossip situation $M'$ such that $M' \approx_c^+ M_1^{ab}$ and $M' = M_1$, for every $(G^{\mathsf{call}+}, M_1^{ab})$ such that $((G, M_1), (G^{\mathsf{call}+}, M_1^{ab})) \in [\![ab^+]\!]$, where $G^{\mathsf{call}+} = (\mathcal{S}^+, \approx^+)$ and $\mathcal{S}^+ = \{N^{cd} \mid N \in \mathcal{S} \text{ and } c \neq d \in \mathsf{Ag}\} \cup \mathcal{S}$; and $N^{cd} = (T, \sim^{N^{cd}}, V)$.

Recall that the initial gossip situation $M_1$ is such that: $\sim_a^{M_1} = \equiv_{\mathsf{Q}_a}^{M_1}$, where $\mathsf{Q}_a = \{A\}$, for all $a \in \mathsf{Ag}$ and where $A$ is the unique secret of agent $a$ (see Definition 3.6). From the semantics of $ab^+$ call in a gossip model (see Definition 3.18), $\mathcal{S}^+ = \mathcal{S}^0 \cup \mathcal{S}$. So the initial gossip situation $M_1$ is in $\mathcal{S}^+$. Now, for every $N^{ab} \in \mathcal{S}^+$ such that $c \notin \{a, b\} \subset \mathsf{Ag}$, we have that $M_1 \approx_c^+ N^{ab}$ (since agent $c$ knows the same secrets in both $M_1$ and $N^{ab}$, see Definition 3.16 and Definition 3.18). Therefore after the $ab^+$ call, agent $c$ considers it possible that the resulting gossip situation is the initial gossip situation. As such:

$$\begin{array}{rcl} G^{\mathsf{call}+}, M_1^{ab} \models \neg K_c \neg \mathit{init} & \iff & G, M_1 \models [ab^+] \neg K_c \neg \mathit{init} \\ & \iff & G \models \mathit{init} \to [ab^+] \neg K_c \neg \mathit{init} \end{array}$$

Consider (‡). We provide a counter example to show that $G \not\models \mathit{init} \to [ab^-] \neg K_c \neg \mathit{init}$. That is, $G^{\mathsf{call}-}, M^{ab} \not\models \neg K_c \neg \mathit{init}$ holds, which is equivalent to saying $G^{\mathsf{call}-}, M^{ab} \models K_c \neg \mathit{init}$ holds. From the semantics of $ab^-$ call in a gossip model (see Definition 3.18), for every $N^{ab} \in \mathcal{S}^-$, $\quad \sim_a^{N^{ab}} = \equiv_{\mathsf{Q}_{ab}}^{N^{ab}}$, where $\sim_a^{N^{ab}} = \sim_a^N \cap \sim_b^N$ and (from Lemma 3.17) $\mathsf{Q}_{ab} = \mathsf{Q}_a \cup \mathsf{Q}_b$. Therefore after the $ab^-$ call, no agent considers it possible that there is a

gossip situation $N^{ab} \in \mathcal{S}^-$ where $\sim_a^{N^{ab}} = \equiv_{\mathsf{Q}_a}^{N^{ab}}$ and $\mathsf{Q}_a = \{A\}$, for all $a \in \mathsf{Ag}$. Therefore:

$$\begin{aligned} G^{\mathsf{call}-}, M_1^{ab} \models K_c \neg init &\iff G, M_1 \models [ab^-] K_c \neg init \\ &\iff G \not\models init \to [ab^-] \neg K_c \neg init \end{aligned}$$

Finally, consider (‡) again. We provide a counter example to show that $G \not\models init \to [ab^0] \neg K_c \neg init$. That is, $G^{\mathsf{call}0}, M_1^{ab} \not\models \neg K_c \neg init$ holds, which is equivalent to saying that $G^{\mathsf{call}0}, M_1^{ab} \models K_c \neg init$ holds. Similar to the foregoing case of $ab^-$, from the semantics of $ab^0$ call in a gossip model, for every $N^{cd} \in \mathcal{S}^0$, $\quad \sim_c^{N^{cd}} = \equiv_{\mathsf{Q}_{cd}}^{N^{cd}}$, where $\sim_c^{N^{cd}} = \sim_c^N \cap \sim_d^N$ and (from Lemma 3.17) $\mathsf{Q}_{cd} = \mathsf{Q}_c \cup \mathsf{Q}_d$. Therefore after the $ab^0$ call, no agent considers it possible that there is a situation $N^{cd} \in \mathcal{S}^0$ where $\sim_c^{N^{cd}} = \equiv_{\mathsf{Q}_c}^{N^{cd}}$ and $\mathsf{Q}_c = \{C\}$, for all $c \in \mathsf{Ag}$. Therefore:

$$\begin{aligned} G^{\mathsf{call}0}, M_1^{ab} \models K_c \neg init &\iff G, M_1 \models [ab^0] K_c \neg init \\ &\iff G \models init \to [ab^0] K_c \neg init \end{aligned}$$

□

*Proof of Proposition 3.22(6).* Let $G = (\mathcal{S}, \approx)$. Let the initial gossip situation be $M_1 = (T, \sim^{M_1}, V)$, where $M_1 \in \mathcal{S}$, then:

$G \models init \to [ab^\mu](K_c \bigvee_{x \neq y} Kw_xY \wedge \neg \bigvee_{x \neq y} K_c Kw_xY)$, only for $\mu \in \{0\}$
$\iff G, M_1 \models [ab^\mu](K_c \bigvee_{x \neq y} Kw_xY \wedge \neg \bigvee_{x \neq y} K_c Kw_xY)$, only for $\mu \in \{0\}$
(assuming the initial gossip situation)

$\iff G^{\mathsf{call}\mu}, M_1^{ab} \models (K_c \bigvee_{x \neq y} Kw_xY \wedge \neg \bigvee_{x \neq y} K_c Kw_xY)$, only for $\mu \in \{0\}$
(from Definition 3.18)

$\iff G^{\mathsf{call}\mu}, M_1^{ab} \models K_c \bigvee_{x \neq y} Kw_xY$ and $G^{\mathsf{call}\mu}, M_1^{ab} \models \neg \bigvee_{x \neq y} K_c Kw_xY$, where $\mu \in \{0\}$
(from the semantics of '$\wedge$')

$\iff G^{\mathsf{call}0}, M_1^{ab} \models K_c \bigvee_{x \neq y} Kw_xY$ and $G^{\mathsf{call}0}, M_1^{ab} \models \neg \bigvee_{x \neq y} K_c Kw_xY$
and $(G^{\mathsf{call}-}, M_1^{ab} \not\models K_c \bigvee_{x \neq y} Kw_xY$ or $G^{\mathsf{call}-}, M_1^{ab} \not\models \neg \bigvee_{x \neq y} K_c Kw_xY)$
and $(G^{\mathsf{call}+}, M_1^{ab} \not\models K_c \bigvee_{x \neq y} Kw_xY$ or $G^{\mathsf{call}+}, M_1^{ab} \not\models \neg \bigvee_{x \neq y} K_c Kw_xY)$
(‡)

for every $(G^{\mathsf{call}\mu}, M_1^{ab})$ such that $((G, M), (G^{\mathsf{call}\mu}, M_1^{ab})) \in [\![ab^\mu]\!]$, and $\mu \in \{-, 0, +\}$, where:

$G^{\mathsf{call}-} = (\mathcal{S}^-, \approx^-)$ and $\mathcal{S}^- = \{N^{ab} \mid N \in \mathcal{S}\}$;
$G^{\mathsf{call}0} = (\mathcal{S}^0, \approx^0)$ and $\mathcal{S}^0 = \{N^{cd} \mid N \in \mathcal{S}$ and $c \neq d \in \mathsf{Ag}\}$;
$G^{\mathsf{call}+} = (\mathcal{S}^+, \approx^+)$ and $\mathcal{S}^+ = \{N^{cd} \mid N \in \mathcal{S}$ and $c \neq d \in \mathsf{Ag}\} \cup \mathcal{S}$;
and $N^{ab} = (T, \sim^{N^{ab}}, V)$, and $N^{cd} = (T, \sim^{N^{cd}}, V)$.

Recall that the initial gossip situation $M_1$ is such that: $\sim_a^{M_1} = \equiv_{\mathsf{Q}_a}^{M_1}$, where $\mathsf{Q}_a = \{A\}$, for all $a \in \mathsf{Ag}$ and where $A$ is the unique secret of agent $a$.

From (‡), let us consider whether it is the case that:

$$G^{\mathsf{call0}}, M_1^{ab} \models K_c \bigvee_{x \neq y} Kw_x Y \text{ and } G^{\mathsf{call0}}, M_1^{ab} \models \neg \bigvee_{x \neq y} K_c Kw_x Y \qquad (‡‡)$$

$G^{\mathsf{call0}}, M_1^{ab} \models K_c \bigvee_{x \neq y} Kw_x Y$ if and only if for every $M'$, $M_1^{ab} \approx_c^0 M'$ implies there is some $\mathsf{Q}$ such that $Y \in \mathsf{Q}$, and some $x \neq y \in \mathsf{Ag}$ such that $\sim_x^{M'} = \equiv_{\mathsf{Q}}^{M'}$. That is, for every $M'$, $M_1^{ab} \approx_c^0 M'$ implies there is some $\mathsf{Q}$ and some $x$ such that $\mathsf{Q} \supset \{X\}$ and $\sim_x^{M'} = \equiv_{\mathsf{Q}}^{M'}$. (††)

That is, for all such $M'$, there is an agent $x$ who knows more secrets than just its own unique secret, that is, agent $x$ knows the unique secret of at least one other agent $y$.

From the semantics of $ab^0$ call in a gossip model (see Definition 3.18), for every $N^{cd} \in \mathcal{S}^0$, $\sim_c^{N^{cd}} = \equiv_{\mathsf{Q}_{cd}}^{N^{cd}}$, where $\sim_c^{N^{cd}} = \sim_c^N \cap \sim_d^N$ and (from Lemma 3.17) $\mathsf{Q}_{cd} = \mathsf{Q}_c \cup \mathsf{Q}_d$. Therefore after the $ab^0$ call, no agent considers it possible that there is a situation $N^{cd} \in \mathcal{S}^0$ where $\sim_c^{N^{cd}} = \equiv_{\mathsf{Q}_c}^{N^{cd}}$ and $\mathsf{Q}_c = \{C\}$, for all $c \in \mathsf{Ag}$. So condition (††) is satisfied since in all the resulting gossip situations due to the $ab^0$ call, some agent knows more secrets than its own unique secret. And therefore:

$$G^{\mathsf{call0}}, M_1^{ab} \models K_c \bigvee_{x \neq y} Kw_x Y$$

Again, from (‡‡),

$$G^{\mathsf{call0}}, M_1^{ab} \models \neg \bigvee_{x \neq y} K_c Kw_x Y \iff G^{\mathsf{call0}}, M_1^{ab} \models \bigwedge_{x \neq y} \neg K_c Kw_x Y$$

To determine whether $G^{\mathsf{call0}}, M_1^{ab} \models \bigwedge_{x \neq y} \neg K_c Kw_x Y$ holds, recall that $c \notin \{a, b\} \subset \mathsf{Ag}$ and that $|Ag| > 3$. Now consider all pairs of gossip situations $M_1^{ab}, M_1^{de} \in \mathcal{S}^0$ such that $c \notin \{a, b\} \cup \{d, e\}$ and $\{a, b\} \neq \{d, e\} \subset \mathsf{Ag}$. Since $M_1$ is the initial gossip situation, it follows that if $\sim_a^{M_1^{ab}} = \equiv_{\mathsf{Q}_{ab}}^{M_1^{ab}}$, and $\sim_d^{M_1^{de}} = \equiv_{\mathsf{Q}_{de}}^{M_1^{de}}$, then $\mathsf{Q}_{ab} \neq \mathsf{Q}_{de}$, for all such pairs $M_1^{ab}, M_1^{de}$ of gossip situations (that is, for all such pairs of gossip situations, each caller in $M_1^{ab}$ learns a different secret from each caller in $M_1^{de}$). But for all such $M_1^{ab}, M_1^{de}$ pairs, $M_1^{ab} \approx_c^0 M_1^{de}$ (agent $c$ was not involved in the calls, and so agent $c$ knows the same secrets in both such situations, and agent $c$ considers all such pairs of gossip situations possible). Therefore after the $ab^0$ call in the initial gossip situation, such agent $c$ considers it possible that there is a gossip situation in which an agent $x$ does not know secret $Y$, $x \neq y \in \mathsf{Ag}$, since agent $c$ considers it possible that such agent $x$ called with agent $y$ at the initial situation, but agent $c$ also considers it possible that agent $x$ called with agent $z \neq y$ at the initial situation $M_1$. And so such agent $c$ does not know the

secret learnt by such agent $x$ after a call at the initial gossip situation. We therefore conclude that: $G^{\mathsf{call0}}, M_1^{ab} \models \bigwedge_{x\neq y} \neg K_c K w_x Y$ holds. And thus (with (††)) we conclude that:

$$G^{\mathsf{call0}}, M_1^{ab} \models K_c \bigvee_{x\neq y} K w_x Y \text{ and } G^{\mathsf{call0}}, M_1^{ab} \models \neg \bigvee_{x\neq y} K_c K w_x Y \qquad (\dagger\dagger\dagger)$$

Next, looking at (‡), we now consider whether:

$$G^{\mathsf{call}-}, M_1^{ab} \not\models K_c \bigvee_{x\neq y} K w_x Y \text{ or } G^{\mathsf{call}-}, M_1^{ab} \not\models \neg \bigvee_{x\neq y} K_c K w_x Y$$

To prove this, it is sufficient to show that:

$$\begin{array}{rcl} G^{\mathsf{call}-}, M_1^{ab} \not\models \neg \bigvee_{x\neq y} K_c K w_x Y & \iff & G^{\mathsf{call}-}, M_1^{ab} \models \bigvee_{x\neq y} K_c K w_x Y \\ & \iff & G^{\mathsf{call}-}, M_1^{ab} \models K_c K w_x Y, \text{ for some } x \neq y \in \mathsf{Ag} \end{array}$$

From the semantics of $ab^-$ call in a gossip model (see Definition 3.18), for every $N^{ab} \in \mathcal{S}^-$, let $\sim_a^{N^{ab}} = \equiv_{\mathsf{Q}_{ab}}^{N^{ab}}$. In this case we are given that the $ab^-$ call was made in the initial situation, so we have that $N = M_1$, and therefore $\sim_a^{M_1^{ab}} = \sim_a^{M_1} \cap \sim_b^{M_1}$, and (from Lemma 3.17) $\mathsf{Q}_{ab} = \mathsf{Q}_a \cup \mathsf{Q}_b$. Therefore in all the gossip situations in $\mathcal{S}^-$, agent $a$ learnt the secret of agent $b$. Therefore, for $x = a$ and $y = b$:

$$\begin{array}{rcl} G^{\mathsf{call}-}, M_1^{ab} \models K_c K w_x Y & \iff & G^{\mathsf{call}-}, M_1^{ab} \models \bigvee_{x\neq y} K_c K w_x Y \\ & \iff & G^{\mathsf{call}-}, M_1^{ab} \not\models K_c \bigvee_{x\neq y} K w_x Y \\ & \text{or} & G^{\mathsf{call}-}, M_1^{ab} \not\models \neg \bigvee_{x\neq y} K_c K w_x Y \end{array} \qquad (\dagger\dagger\dagger\dagger)$$

Finally, looking at (‡), we now consider whether:

$$G^{\mathsf{call}+}, M_1^{ab} \not\models K_c \bigvee_{x\neq y} K w_x Y \text{ or } G^{\mathsf{call}+}, M_1^{ab} \not\models \neg \bigvee_{x\neq y} K_c K w_x Y$$

To prove this, it is sufficient to show that:

$$G^{\mathsf{call}+}, M_1^{ab} \not\models K_c \bigvee_{x\neq y} K w_x Y \iff G^{\mathsf{call}+}, M_1^{ab} \models \neg K_c \bigvee_{x\neq y} K w_x Y$$

From the semantics of $ab^+$ call in a gossip model (see Definition 3.18), $\mathcal{S}^+ = \mathcal{S}^0 \cup \mathcal{S}$. So the initial gossip situation $M_1$ is in $\mathcal{S}^+$. Now, for every $N^{ab} \in \mathcal{S}^+$ such that $c \notin \{a, b\} \subset \mathsf{Ag}$, we have that $M_1 \approx_c^+ N^{ab}$ (from Definition 3.7 and Definition 3.18). Therefore after the $ab^+$ call, agent $c$ considers it possible that the resulting situation is the initial gossip situation $M_1$. As such, for any given $x \neq y \in \mathsf{Ag}$:

$$\begin{array}{rcl} G^{\mathsf{call}+}, M_1^{ab} \models \neg K_c \bigvee_{x\neq y} K w_x Y & \iff & G^{\mathsf{call}+}, M_1^{ab} \not\models K_c \bigvee_{x\neq y} K w_x Y \\ & \iff & G^{\mathsf{call}+}, M_1^{ab} \not\models K_c \bigvee_{x\neq y} K w_x Y \\ & \text{or} & G^{\mathsf{call}+}, M_1^{ab} \not\models \neg \bigvee_{x\neq y} K_c K w_x Y \end{array} \qquad (\dagger\dagger\dagger\dagger\dagger)$$

In conclusion, from (†††), (††††) and (†††††), we conclude that (‡) holds. □

Not every gossip model can be the result of a sequence of calls from the initial gossip model. For instance, it is not possible to reach a gossip situation (or a gossip model containing such a gossip situation) in which agent $a$ knows everybody's secret, but all other agents only know their own secret. Furthermore, the execution of $ab^-$ calls on the initial gossip model preserves the property that it is common knowledge who knows which secrets. In other words, after an $ab^-$ call, consider an agent $c$ and a secret $D$. For every other agent $e$, agent $c$ knows if agent $e$ knows whether $D$. This property obviously does not hold for $ab^0$ calls. Finally, there is no common knowledge of information growth after $ab^+$ calls. After an $ab^+$ call, an agent $c \neq a, b$ considers it possible that no call was made at all.

**A Discussion on Calls as Action Models.** The different call actions can also be considered as a special case of action models described in Section 2.2. In this special case, the $ab^\mu$ construct in Definitions 3.1 and 3.2 will not be considered as a single action point from the domain of an action model (as in Definition 2.10), but instead call $ab^\mu$ will be considered as the set of all the action points in the action model. Hence there will be no designated action point in an action model for call $ab^\mu$. And thus we allow that the call $ab^\mu$ can be made under any one of all the possible preconditions for such a call.

In our gossip scenarios, when the agents $a$ and $b$ call each other, they exchange all the secrets they know. So, both agents can distinguish calls wherein either of them knows a different set of secrets. So, to determine all the possible preconditions for a call between agent $a$ and $b$, we need to list for each agent : (i) the secrets that it knows to be true, (ii) the secrets that it knows to be false, and (iii) the secrets it does not know. Those in (i) and (ii) are in non-overlapping subsets and (iii) can be their complement.

Given the $n$ secrets $\mathsf{P}$, agent $a$ may currently know that the secrets in $\mathsf{Q}_a^+ \subseteq \mathsf{P}$ are true and those in $\mathsf{Q}_a^- \subseteq \mathsf{P}$ are false (and suppose $\mathsf{Q}_a^+ \cup \mathsf{Q}_a^- = \mathsf{Q}_a$), and be ignorant about the rest; whereas agent $b$ may currently know that the secrets in $\mathsf{Q}_b^+ \subseteq \mathsf{P}$ are true and that those in $\mathsf{Q}_b^- \subseteq \mathsf{P}$ are false (and we let $\mathsf{Q}_b^+ \cup \mathsf{Q}_b^- = \mathsf{Q}_b$). We now define:

$$\begin{array}{rcl} \delta(\mathsf{Q}_a^+, \mathsf{Q}_a^-, \mathsf{Q}_b^+, \mathsf{Q}_b^-) & ::= & \bigwedge_{C \in \mathsf{Q}_a^+} K_a C \wedge \bigwedge_{C \in \mathsf{Q}_a^-} K_a \neg C \wedge \\ & & \bigwedge_{C \in \mathsf{P} \setminus \mathsf{Q}_a} Ig_a C \wedge \\ & & \bigwedge_{C \in \mathsf{Q}_b^+} K_b C \wedge \bigwedge_{C \in \mathsf{Q}_b^-} K_b \neg C \wedge \\ & & \bigwedge_{C \in \mathsf{P} \setminus \mathsf{Q}_b} Ig_b C \end{array}$$

where $Ig_a C = \neg K w_a C,\ a \in \mathsf{Ag}$.

This formula $\delta(\mathsf{Q}_a^+, \mathsf{Q}_a^-, \mathsf{Q}_b^+, \mathsf{Q}_b^-)$ is a *precondition* of an action point in the domain of any of the action models for call $ab^\mu$.

For call $ab^-$, all agents know that the call between $a$ and $b$ takes place. They only do not know the value of the exchanged secrets. Thus the action model for call $ab^-$ consists of a domain containing different actions points for all preconditions of type

$\delta(\mathsf{Q}_a^+, \mathsf{Q}_a^-, \mathsf{Q}_b^+, \mathsf{Q}_b^-)$. Agents other than $a, b$ have the universal accessibility relation on this action model and agents $a, b$ have identity accessibility relation. The action model satisfies that all actions are mutually exclusive and that the union of all preconditions is the trivial formula. So always exactly one action fires, and the result is a refinement of the gossip model (no states are eliminated or duplicated, it is merely the case that the accessibility relations for the agents $a$ and $b$ are more refined).

Still considering call $ab^-$, let is pause briefly to estimate the number of different such preconditions $\delta$. For each agent $x \in \{a, b\}$ and for each secret $Y \in \mathsf{P}$, we consider three possible atomic propositions, namely: $K_xY$, $K_x\neg Y$, and $Ig_xY$. So for agents $a$ and $b$, there are $n$ placeholders each, where $n$ is the number of secrets in the scenario, and each of those placeholders can take any of the three foregoing atomic propositions. So estimating the number of different such formulas of type $\delta$ can be seen as estimating the number of all possible states given by a ternary word of size $2n$. Hence an upper bound will be $O(3^{2n})$. Simplifications are possible, for example, we may require that $A \in \mathsf{Q}_a$ and $B \in \mathsf{Q}_b$; but this does not simplify matters greatly. This simplification leads us to restrict two placeholders, out of the said $2n$ placeholders, to having only two possible 'values' (that is, we can no longer have a formula of $\delta$ with $Ig_aA$ and $Ig_bB$ conjuncts). So the upper bound on the number of such formulas comes down to $O(3^{2n} \times 2/3 \times 2/3) = O(3^{2n-2} \times 4) = O(3^{2n-2})$ different such preconditions $\delta$ for the simplified action model, and such a simplified action model would have the same update effect on gossip models.

Finally, for call $ab^0$, we obtain the action points in the action model in the same way as for call $ab^-$, but now instead of considering only all the different action points corresponding to all the preconditions for agent $a$ and $b$ to make a call, we consider all the different action points corresponding to all the preconditions for any pair of agents to make a call. Thus, given $n$ agents in the scenario, we obtain $n(n-1)$ times the number of action points in the model of for call $ab^-$. To obtain the action points in the action model for call $ab^+$, we add a 'no call happens' action point (with precondition *true*) to the domain of the action model for call $ab^0$.

## 3.6 Formalising Epistemic Gossip Protocols

From now on, we consider calls $ab^0$ only, written simply as $ab$. Our definitions will equally apply to protocols with $ab^-$ and $ab^+$ calls, but with minor adaptations.

Our epistemic protocols should be seen on a par with *knowledge programs* [22] and, more broadly with *epistemic planning* [10]. The idea is that every agent has its own program where the actions chosen by the agent are conditional on what the agent *knows*. In our setting, one might expect that the appropriate pre-condition to make a call is ignorance rather than knowledge, but the reader is reminded that in epistemic logic, the ignorance condition $\neg K_a\varphi$ is equivalent to the knowledge condition $K_a\neg K_a\varphi$. We assume that an individual agent program of agent $a$ specifies under which conditions $a$

would like to make a phone call, and to what kind of partner. In case conditions for different agents apply, an arbiter will chose whose request is granted.

The Fixed Schedule is not an epistemic protocol: the agents appearing in the protocol are *names* and not *variables*, and the actions are selected independently of what an agent knows.

So, towards epistemic gossip protocols, we assume sets of variables $\{x, y, z, \dots\}$ and $\{X, Y, Z, \dots\}$ over agents and secrets. We consider a language $\mathcal{L}_\Pi$ for protocols which is obtained from Definition 3.2 by replacing $A$ by $X$, $a$ by $x$ and $b$ by $y$. Define the free variables $FV(X) = \{x\}$, $FV(K_x\varphi) = \{x\} \cup FV(\varphi)$, and $FV([xy]\varphi) = \{x, y\} \cup FV(\varphi)$. Moreover, $FV(\neg\varphi) = FV(\varphi)$ and $FV(\varphi_1 \wedge \varphi_2) = FV(\varphi_1) \cup FV(\varphi_2)$. We also allow the following constructs in the language, with the associated free variables for them:

$$FV(\bigwedge_{z\in\mathsf{Ag}} \varphi) = FV(\bigvee_{z\in\mathsf{Ag}} \varphi) = FV(\cup_{z\in\mathsf{Ag}}\varphi) = FV(\varphi) \setminus \{z\}$$

We say that $\psi$ is about $x$ and $y$, and write $\psi(x, y)$ if $FV(\psi) = \{x, y\}$. As an example, take $\psi(x, y) = K_x \bigvee_{z\in\mathsf{Ag}}(Kw_yZ \wedge \neg Kw_xZ)$ ($x$ knows that $y$ knows a secret that $x$ does not know).

**Definition 3.23** (Epistemic gossip protocol)**.** To define an epistemic gossip protocol $\Pi$, we assume $\psi(x, y) \in \mathcal{L}_{Kw}$ to be a formula about $x$ and $y$. We then define for every $\Pi$ an *epistemic calling condition* (for $x$ to call $y$) $cc(x, y, \Pi)$ as:

$$cc(x, y, \Pi) = K_x\psi(x, y) \tag{3.3}$$

An *epistemic gossip protocol* $\Pi$ is then a program of the form:

$$\texttt{while} \bigvee_{x,y\in\mathsf{Ag}} cc(x, y, \Pi) \texttt{ do } \bigcup_{x,y\in\mathsf{Ag}} (?cc(x, y, \Pi)\,; xy) \tag{3.4}$$

In words: as long as there are two agents $x$ and $y$ for which the condition is true, choose such a pair and let them make the call. Less restrictive definitions of protocols are definitely plausible: the termination condition might be different from the epistemic calling condition, and the epistemic calling condition might be different for different agents, for example. Since $(K_x\psi_1 \vee K_x\psi_2)$ is equivalent to $K_x(K_x\psi_1 \vee K_x\psi_2)$, our definition does allow for test which are based on cases.

**Definition 3.24** (Execution sequence)**.** Given the pointed gossip model $(G, M_1)$, where $M_1$ is the initial gossip situation. Let $\mu$ be the mode of call. Then we inductively define an *execution sequence* $\sigma_\Pi$ of an epistemic gossip protocol $\Pi$ as follows:

**Base Case**

- $ab$ is an execution sequence of $\Pi$ if and only if, where $\mu \in \{-, 0, +\}$:
  - $G, M_1 \models cc(a, b, \Pi)$, and,

– there is some $(G^{\mathsf{call}\mu}, M^{ab})$ such that $(G, M_1)[\![ab]\!](G^{\mathsf{call}\mu}, M^{ab})$.

- $\mathsf{skip}$ is an execution sequence of $\Pi$ if and only if:
  – the call mode $\mu = +$, and,
  – $G, M_1 \models cc(x, y, \Pi)$ for some $x, y \in \mathsf{Ag}$, and,
  – $(G, M_1)[\![\mathsf{skip}]\!](G, M_1)$.

**Inductive Case**

- $\sigma_\Pi; ab$ is an execution sequence of $\Pi$ if and only if, where $\mu \in \{-, 0, +\}$:
  – $\sigma_\Pi$ is an execution sequence of $\Pi$, and
  – $G^{\mathsf{call}\mu}, M^{\sigma_\Pi} \models cc(a, b, \Pi)$, for all $(G^{\mathsf{call}\mu}, M^{\sigma_\Pi})$ such that $(G, M_1)\ [\![\sigma_\Pi]\!]\ (G^{\mathsf{call}\mu}, M^{\sigma_\Pi})$, and,
  – there is some $(G^{\mathsf{call}\mu}, M^{\sigma_\Pi; ab})$ such that $(G, M_1)\ [\![\sigma_\Pi; ab]\!]\ (G^{\mathsf{call}\mu}, M^{\sigma_\Pi; ab})$.
- $\sigma_\Pi; \mathsf{skip}$ is an execution sequence of $\Pi$ if and only if, where $\mu = +$:
  – $\sigma_\Pi$ is an execution sequence of $\Pi$, and
  – for some $x, y \in \mathsf{Ag}$, $G^{\mathsf{call}\mu}, M^{\sigma_\Pi} \models cc(x, y, \Pi)$, for all $(G^{\mathsf{call}\mu}, M^{\sigma_\Pi})$ such that $(G, M_1)[\![\sigma_\Pi]\!](G^{\mathsf{call}\mu}, M^{\sigma_\Pi})$, and
  – $(G, M_1)[\![\sigma_\Pi; \mathsf{skip}]\!](G^{\mathsf{call}\mu}, M^{\sigma_\Pi})$.

Notice that an execution sequence[§] always begins at the initial gossip situation. The $\mathsf{skip}$ action is allowed only in a $+$-mode call, and only when there is some pair of agents for which the epistemic calling condition of the protocol is satisfied. In other words, in the $+$-mode, the $\mathsf{skip}$ action could be executed even though a call between two agents is possible.

**Definition 3.25** (Extension of an Epistemic Gossip Protocol)**.** The *extension* $\Sigma(\Pi)$ of an epistemic gossip protocol $\Pi$ is the set of all its execution sequences of calls.

The gossip situation sequences of an epistemic gossip protocol $\Pi$ are all the sequences of gossip situations generated by $\Pi$. If protocols have the same extension, they obviously have the same meaning and gossip situation sequences. But protocols may have the same gossip situation sequences and still have different extensions: obviously the two call sequences $ab; ac; ab$ and $ab; ac; bc$ are different, yet they generate the same gossip situation sequences, that is, $A.B.C \to AB.AB.C \to ABC.AB.ABC \to ABC.ABC.ABC$.

We now present some examples. Since a protocol is completely determined by its condition $cc(x, y, \Pi)$, we only give those conditions for the protocols here. Obviously, there is a connection between the logical strength of this condition for $\Pi$ and the set of its extension.

**Proposition 3.26.** *For any protocols* $\Pi$ *and* $\Pi'$*,*

$$\models_g cc(x, y, \Pi) \to cc(x, y, \Pi') \text{ implies } \Sigma(\Pi) \subseteq \Sigma(\Pi')$$

[§]Note that we use the term *execution sequence* and *call sequence* interchangeably.

*Proof.* Take any gossip model $G$, and consider an arbitrary situation $M$ in such a model. Suppose that $G, M \models cc(x, y, \Pi) \rightarrow cc(x, y, \Pi')$. Then it follows that if a call $xy$ can be made for protocol $\Pi$, then the same call can also be made for protocol $\Pi'$, in the situation $M$. Since the foregoing statement is true for an arbitrary situation $M$ in $G$, it follows that any execution sequence in $\Sigma(\Pi)$ is also in $\Sigma(\Pi')$. And therefore $\Sigma(\Pi) \subseteq \Sigma(\Pi')$. □

The definition of execution sequence given in Definition 3.24 considers that the protocol in use in the scenario may not be common knowledge among all the agents, and, different agents may use different protocols in the same scenario. As a result, Definition 3.24, for the 0-mode calls, is based on the more generic model $G^{\mathsf{call0}}$, rather than the model of any specific protocol. Note that if it is not common knowledge that a given protocol is in use by all the agents in a gossip scenario, then the calls that are not allowed by that protocol at a given gossip situation could still be possible at that gossip situation for various reasons, e.g., another agent considers such a call possible because it does not know the protocol used by the calling pair, or, the call is allowed by the protocol used by the agent who initiated the call in question.

In order to make a given protocol common knowledge to all agents, we need to slightly adjust the semantics of calls, that is, for each protocol $\Pi$, we have to replace $\mathcal{S}^0$ of Definition 3.18 by:

$$\mathcal{S}^0_\Pi = \{N^{cd} \mid N \in \mathcal{S} \;\&\; G, N \models cc(c, d, \Pi)\} \tag{3.5}$$

Syntactically, to make a given protocol common knowledge to all agents, we need to restrict the language: $\mathcal{L}_K(\Pi)$ is obtained by adapting the object language $\mathcal{L}_K$ in such a way, that the only program $\pi$ that occurs is the program of the form (3.4).

### 3.6.1 The Epistemic Calling Conditions of the Sample Protocols

We now define the epistemic calling condition for each of our epistemic gossip protocols, as follows.

*Protocol* 1 (Learn New Secrets). We define the epistemic calling condition for Learn New Secrets $\Pi_1$ as follows:

$$cc(x, y, \Pi_1) = K_x \neg K w_x Y \tag{3.6}$$

The condition for $x$ to call $y$ in $\Pi_1$ in words is simple: $x$ calls any agent whose secret it yet does not know. The minimum length of a call sequence for this protocol is $2n-4$ and the maximum length is $n(n-1)/2$ (see, later, Chapter 6 for proofs). For the minimum, consider the following sequence, which is a variant of the Fixed Schedule: fix four different agents $a, b, c, d$ from $\mathsf{Ag}$. First, $a$ makes $n-4$ calls to all $\mathsf{Ag} \setminus \{a, b, c, d\}$. Then, the calls $ab; cd; ac; bd$ are made. Finally all agents from $\mathsf{Ag} \setminus \{a, b, c, d\}$ call agent $b$. For the maximum, the sequence that constitutes Equation (3.7) is an example.

$$\sigma = a_1a_2; a_1a_3; \ldots; a_1a_n; a_2a_3; \ldots; a_2a_n; a_3a_4; \ldots; a_{n-1}a_n \tag{3.7}$$

*Protocol* 2 (Known Information Growth de Dicto). We define the epistemic calling condition for Known Information Growth de Dicto $\Pi_2$ as follows:

$$cc(x, y, \Pi_2) = K_x(\bigvee_{z \in \mathsf{Ag}} Kw_x Z \ \nabla \ Kw_y Z) \tag{3.8}$$

Here, $\nabla$ denotes exclusive or. In order for $x$ to call $y$, condition $cc(x, y, \Pi_2)$ requires that $x$ should know that some secret is currently known by only one of $x$ and $y$: So, $x$ will call $y$ if $x$ knows this call will produce new knowledge of secrets.

*Protocol* 3 (Known Information Growth de Re). We define the epistemic calling condition for Known Information Growth de Re $\Pi_3$ as follows:

$$cc(x, y, \Pi_3) = K_x \bigvee_{z \in \mathsf{Ag}} K_x(Kw_x Z \ \nabla \ Kw_y Z) \tag{3.9}$$

Contrast $cc(x, y, \Pi_2)$ with $cc(x, y, \Pi_3)$: under the latter, $x$ is allowed to call $y$ if there is some secret $Z$ of which $x$ knows that only one of $x$ and $y$ knows it. The condition $cc(x, y, \Pi_2)$ is a knowledge *de Dicto* requirement. The epistemic calling condition $cc(x, y, \Pi_3)$ shows that our language also allows for a knowledge *de Re* condition.

Note that this condition is equivalent to $\bigvee_{z \in \mathsf{Ag}} K_x(Kw_x Z \ \nabla \ Kw_y Z)$. To appreciate the difference between the two Known Information Growth protocols, suppose we have four agents and a call sequence starting with $\sigma = ab; bc; cd$. After this sequence, $cc(a, b, \Pi_2)$ holds ($a$ knows it was not involved in the last two calls, so $b$ must have learned something new), but $cc(a, b, \Pi_3)$ does not ($a$ does not know *what* $b$ has learned). However, after $\sigma; bd$, agent $a$ *does* know that $b$ must have learned $C$, and so now both $cc(a, b, \Pi_2)$ and $cc(a, b, \Pi_3)$ are true. This also demonstrates a difference between Learn New Secrets and the Known Information Growth protocols: whereas the Known Information Growth protocols allow for two agents $a$ and $b$ to call each other more than once in a call sequence, Learn New Secrets protocol does not.

On the one hand, the condition for Known Information Growth (unless explicitly specified, we assume *de Dicto* versions of protocols) assumes a cooperative agent $x$: even if it knows that only agent $y$ will benefit from the call, agent $x$ will make it. However, on the other hand those conditions may look rather strong: under certain circumstances, it may be reasonable for agent $x$ to call agent $y$ even if agent $x$ is not sure this will result in growth of information. Let us write $\hat{K}_a\varphi$ for $\neg K_a \neg \varphi$, that is, for agent $a$, $\varphi$ is an epistemic possibility. In standard epistemic logic, $K_a\hat{K}_a\varphi \leftrightarrow \hat{K}_a\varphi$ (for '$\Rightarrow$', use veridicality; for '$\Leftarrow$', observe that $\hat{K}_x\varphi$ implies $K_x\hat{K}_x\varphi$, by negative introspection). We now define the two final epistemic protocols.

*Protocol* 4 (Possible Information Growth de Dicto). We define the epistemic calling condition for Possible Information Growth de Dicto $\Pi_4$ as follows:

$$cc(x, y, \Pi_4) = K_x\hat{K}_x(\bigvee_{z \in \mathsf{Ag}} Kw_x Z \ \nabla \ Kw_y Z) \tag{3.10}$$

*Protocol* 5 (Possible Information Growth de Re). We define the epistemic calling condition for Possible Information Growth de Re $\Pi_5$ as follows:

$$cc(x, y, \Pi_5) = K_x \bigvee_{z \in \mathsf{Ag}} K_x \hat{K}_x (Kw_x Z \nabla Kw_y Z) \tag{3.11}$$

In words, $x$ is allowed to call $y$, if, according to $cc(x, y, \Pi_4)$, $x$ considers it possible that some secret becomes shared knowledge by such a call. ($\Pi_5$ is the *de Re* variant: note that $cc(x, y, \Pi_5)$ is equivalent to $\bigvee_{z \in \mathsf{Ag}} \hat{K}_x (Kw_x Z \nabla Kw_y Z)$).

**Proposition 3.27.** *Let* $\Pi_0$ *denote the Fixed Schedule:*

1. $\Sigma(\Pi_1) \subsetneq \Sigma(\Pi_3) \subsetneq \Sigma(\Pi_2) \subsetneq \Sigma(\Pi_5) = \Sigma(\Pi_4)$
2. $\Sigma(\Pi_0) \not\subseteq \Sigma(\Pi_1)$ *and* $\Sigma(\Pi_0) \subsetneq \Sigma(\Pi_3)$ *and* $\Sigma(\Pi_0) \subsetneq \Sigma(\Pi_2)$ *and* $\Sigma(\Pi_0) \subsetneq \Sigma(\Pi_4)$ *and* $\Sigma(\Pi_0) \subsetneq \Sigma(\Pi_5)$

*Proof of Proposition 3.27(1).* Making use of the Proposition 3.26, we show that Proposition 3.27(1) is the case by arguing for the following claims:

**Claim 1:** $\models_g cc(x, y, \Pi_1) \to cc(x, y, \Pi_3)$**.**

This claim says that for every gossip model $G'$, and for every gossip situation $M'$ in the domain of $G'$, that: $G', M' \models cc(x, y, \Pi_1) \to cc(x, y, \Pi_3)$. Now let $G$ be an arbitrary gossip model, and let $M$ be an arbitrary gossip situation in the domain of $G$. Then:

$$G, M \models cc(x, y, \Pi_1) \to cc(x, y, \Pi_3) \iff$$
$$G, M \models K_x \neg Kw_x Y \to K_x \bigvee_{z \in \mathsf{Ag}} K_x (Kw_x Z \nabla Kw_y Z), \text{ for some } x, y, z \in \mathsf{Ag} \iff$$
$$G, M \models K_x \neg Kw_x Y \to K_x \bigvee_{z \in \mathsf{Ag}} K_x (\ (Kw_x Z \wedge \neg Kw_y Z) \ \vee \ (\neg Kw_x Z \wedge Kw_y Z)\ ) \iff$$
$$G, M \models K_x \neg Kw_x Y \to K_x \bigvee_{z \in \mathsf{Ag}} K_x (Kw_x Z \wedge \neg Kw_y Z) \ \vee K_x (\neg Kw_x Z \wedge Kw_y Z) \iff$$
$$G, M \models K_x \neg Kw_x Y \to K_x \bigvee_{z \in \mathsf{Ag}} (K_x Kw_x Z \wedge K_x \neg Kw_y Z) \ \vee (K_x \neg Kw_x Z \wedge K_x Kw_y Z).$$

Now suppose $G, M \models K_x \neg Kw_x Y$, where $y \in \mathsf{Ag}$. But also, agent $x$ knows that $Kw_y Y$. Therefore:

$$G, M \models K_x \neg Kw_x Y \wedge K_x Kw_y Y \iff G, M \models K_x (K_x \neg Kw_x Y \wedge K_x Kw_y Y).$$

**Claim 2:** $\not\models_g cc(x, y, \Pi_3) \to cc(x, y, \Pi_1)$**.**

To show that this claim is true, consider a gossip model with three agents $x, y, z$. Assume a gossip situation in which agent $x$ and agent $y$ know each other's secret. Furthermore, assume that agent $x$ knows that only one of agent $x$ and agent $y$ knows the secret of agent $z$, where $z \neq x \neq y$ ($cc(x, y, \Pi_3)$ is satisfied). Finally,

suppose that agent $x$ knows the secret of agent $z$. Now we see that although $cc(x, y, \Pi_3)$ is satisfied, $cc(x, y, \Pi_1)$ is not satisfied since agent $x$ already know the secret of agent $y$.

**Claim 3:** $\models_g cc(x, y, \Pi_3) \rightarrow cc(x, y, \Pi_2)$**.**

This claim says that for every gossip model $G'$, and for every gossip situation $M'$ in the domain of $G'$, that: $G', M' \models cc(x, y, \Pi_3) \rightarrow cc(x, y, \Pi_2)$. Now let $G$ be an arbitrary gossip model, and let $M$ be an arbitrary gossip situation in the domain of $G$. Then:

$$\models_g cc(x, y, \Pi_3) \rightarrow cc(x, y, \Pi_2) \Longleftrightarrow$$
$$G, M \models K_x \bigvee_{z \in \mathsf{Ag}} K_x(Kw_xZ \nabla Kw_yZ) \rightarrow K_x(\bigvee_{z \in \mathsf{Ag}} Kw_xZ \nabla Kw_yZ) \quad (3.12)$$

Suppose:

$$G, M \models K_x \bigvee_{z \in \mathsf{Ag}} K_x(Kw_xZ \nabla Kw_yZ) \quad (3.13)$$

We want to show that:

$$G, M \models K_x(\bigvee_{z \in \mathsf{Ag}} Kw_xZ \nabla Kw_yZ) \quad (3.14)$$

Note that in $S5$, $K_x(K_x\alpha \vee K_x\beta)$ is equivalent to $(K_x\alpha \vee K_x\beta)$ (for '$\Rightarrow$', use veridicality, for '$\Leftarrow$', observe that $K_x\alpha$ implies $K_xK_x\alpha$ which in turn implies $K_x(K_x\alpha \vee K_x\beta)$). Likewise, $K_x\beta$ implies $K_x(K_x\alpha \vee K_x\beta)$, from which the result follows.

Now let $\mathsf{Ag} = \{a_1, \ldots, a_n\}$, where $x = a_i$, $y = a_j$ and $z = a_k$. Let $\alpha_r = (Kw_xA_r \nabla Kw_yA_r)$, where $1 \leq r \leq n$. Therefore Claim (3.13) is equivalent to:

$$G, M \models K_x(K_x\alpha_1 \vee \cdots \vee K_x\alpha_n) \Longleftrightarrow G, M \models (K_x\alpha_1 \vee \cdots \vee K_x\alpha_n) \quad (3.15)$$

And from Claim (3.15), $G, M \models (K_x\alpha_1 \vee \cdots \vee K_x\alpha_n)$ implies $G, M \models K_x(\alpha_1 \vee \cdots \vee \alpha_n)$, which in turn is equivalent to Claim (3.14).

**Claim 4:** $\not\models_g cc(x, y, \Pi_2) \rightarrow cc(x, y, \Pi_3)$**.**

Consider a gossip model with four agents $w, x, y, z$. Assume that $M$ is the actual gossip situation, and that $M$ is as a result of the following: agent $x$ calls agent $w$ at the initial gossip situation (so $w$ and $x$ now know each other's secret), then there are two further calls in which agent $x$ was not involved. Certainly in $M$, agent $x$ knows that it will learn some secret by calling agent $w$ (since $w$ must have called with at least one of the other agents in the two intervening calls since the last time it called with $x$) ($cc(x, y, \Pi_2)$ is satisfied). However, agent $x$ is uncertain as to which particular secret it will learn by calling agent $w$ again (since the secrets agent

$w$ now knows depends on the sequence of two calls that actually took place since the first $xw$ call, and then agent $x$ is uncertain about what this actual sequence is) ($cc(x, y, \Pi_3)$ is not satisfied).

**Claim 5:** $\models_g cc(x, y, \Pi_2) \rightarrow cc(x, y, \Pi_5)$**.**

This claim says that for every gossip model $G'$, and for every gossip situation $M'$ in the domain of $G'$, that: $G', M' \models cc(x, y, \Pi_2) \rightarrow cc(x, y, \Pi_5)$. Now let $G$ be an arbitrary gossip model, and let $M$ be an arbitrary gossip situation in the domain of $G$. Then:

$$\models_g cc(x, y, \Pi_2) \rightarrow cc(x, y, \Pi_5) \Longleftrightarrow$$
$$G, M \models K_x(\bigvee_{z \in \mathsf{Ag}} Kw_x Z \nabla Kw_y Z) \rightarrow K_x \bigvee_{z \in \mathsf{Ag}} K_x \hat{K}_x (Kw_x Z \nabla Kw_y Z).$$

Suppose $G, M \models K_x(\bigvee_{z \in \mathsf{Ag}} Kw_x Z \nabla Kw_y Z)$. That is,

$$\text{for every } M', \quad M \approx_x M' \text{ implies } G, M' \models \bigvee_{z \in \mathsf{Ag}} (Kw_x Z \nabla Kw_y Z) \quad (3.16)$$

$$\text{We want to show that: } G, M \models K_x \bigvee_{z \in \mathsf{Ag}} K_x \hat{K}_x (Kw_x Z \nabla Kw_y Z) \quad (3.17)$$

Note that in $S5$, $K_x(K_x\alpha \vee K_x\beta)$ is equivalent to $(K_x\alpha \vee K_x\beta)$ (for '$\Rightarrow$', use veridicality; for '$\Leftarrow$', observe that $K_x\alpha$ implies $K_xK_x\alpha$ which in turn implies $K_x(K_x\alpha \vee K_x\beta)$). Likewise, $K_x\beta$ implies $K_x(K_x\alpha \vee K_x\beta)$, from which the result follows.

Now let $\mathsf{Ag} = \{a_1, \ldots, a_n\}$, where $x = a_i$, $y = a_j$ and $z = a_k$. Let $\alpha_r = \hat{K}_x(Kw_x A_r \nabla Kw_y A_r)$, where $1 \leq r \leq n$. Therefore Claim (3.17) is equivalent to:

$$G, M \models K_x(K_x\alpha_1 \vee \cdots \vee K_x\alpha_n) \Longleftrightarrow G, M \models (K_x\alpha_1 \vee \cdots \vee K_x\alpha_n) \quad (3.18)$$

Again, to show that Claim (3.18) is true: '$\Rightarrow$' follows from veridicality; to show that '$\Leftarrow$' holds, consider that $K_x\alpha_r$ implies $K_xK_x\alpha_r$ (by positive introspection), which in turn implies $K_x(K_x\alpha_1 \vee \cdots \vee K_x\alpha_n)$ (by propositional logic), for all $r \in \{1, \ldots, n\}$.

Now let $\alpha'_r = (Kw_x A_r \nabla Kw_y A_r)$, where $1 \leq r \leq n$. So from Claim (3.18),

$$\begin{array}{rcl} G, M \models (K_x\alpha_1 \vee \cdots \vee K_x\alpha_n) & \Longleftrightarrow & G, M \models K_x\hat{K}_x\alpha'_1 \vee \cdots \vee K_x\hat{K}_x\alpha'_n) \\ & \Longleftrightarrow & G, M \models \hat{K}_x\alpha'_1 \vee \cdots \vee \hat{K}_x\alpha'_n \end{array} \quad (\ddagger)$$

(for '$\Rightarrow$', use veridicality; for '$\Leftarrow$', observe that $\hat{K}_x\alpha'$ implies $K_x\hat{K}_x\alpha'$, by negative introspection).

But from the semantics of '$\hat{K}_x$' operator, Claim ($\ddagger$) holds if and only if there is some $M'$ such that $M \approx_x M'$ and $G, M' \models \alpha'_1 \vee \cdots \vee \alpha'_n$. That is, there is some $M'$

such that $M \approx_x M'$ and $G, M' \models (Kw_xA_1 \nabla Kw_yA_1) \vee \cdots \vee (Kw_xA_n \nabla Kw_yA_n)$. And this is true, based on Assumption (3.16).

**Claim 6:** $\not\models_g cc(x, y, \Pi_5) \rightarrow cc(x, y, \Pi_2)$**.**

Consider a gossip scenario with four agents $w, x, y, z$. Assume that $M$ is the actual gossip situation, and that $M$ is as a result of the following execution sequence: $xy; xw; wy$. Now, in $M$, agent $x$ considers it possible that it will learn the secret of agent $z$ by calling agent $y$ since agent $x$ considers it possible that the third call in the sequence of calls that gave rise to $M$ was in fact call $yz$, in which agent $y$ must have learnt the secret of agent $z$ (since agent $x$ was not involved in the third call, agent $x$ is uncertain as to which call actually occurred in the third round). Notice from the foregoing that $cc(x, y, \Pi_5)$ is satisfied at $M$. (Note that agent $x$ cannot distinguish the gossip situations due to the two execution sequences $xy; xw; wy$ and $xy; xw; yz$ since, from Definition 3.18, agent $x$ knows the same set of secrets in the resulting gossip situation due to each of the given two execution sequences). On the other hand however, agent $x$ does not *know* if it (agent $x$) or agent $y$ will learn any new secret by calling each other in $M$, since agent $x$ considers it possible that the execution sequence that gave rise to $M$ is $xy; xw; wy$, in which case neither agent $x$ nor agent $y$ will have any new secret to tell each other in a fourth round call between them. Thus $cc(x, y, \Pi_2)$ is *not* satisfied in $M$.

**Claim 7:** $\models_g cc(x, y, \Pi_5) \leftrightarrow cc(x, y, \Pi_4)$**.**

This claim says that for every gossip model $G'$, and for every gossip situation $M'$ in the domain of $G'$, that: $G', M' \models cc(x, y, \Pi_5) \leftrightarrow cc(x, y, \Pi_4)$. Now let $G$ be an arbitrary gossip model, and let $M$ be an arbitrary gossip situation in the domain of $G$. Then:

$$\models_g cc(x, y, \Pi_5) \leftrightarrow cc(x, y, \Pi_4) \Longleftrightarrow$$
$$G, M \models K_x \bigvee_{z \in \mathsf{Ag}} K_x\hat{K}_x(Kw_xZ \nabla Kw_yZ) \leftrightarrow K_x\hat{K}_x(\bigvee_{z \in \mathsf{Ag}} Kw_xZ \nabla Kw_yZ)$$

Note that in $S5$, $K_x(K_x\alpha \vee K_x\beta)$ is equivalent to $(K_x\alpha \vee K_x\beta)$ (for '$\Rightarrow$', use veridicality, for '$\Leftarrow$', observe that $K_x\alpha$ implies $K_xK_x\alpha$ which in turn implies $K_x(K_x\alpha \vee K_x\beta)$). Likewise, $K_x\beta$ implies $K_x(K_x\alpha \vee K_x\beta)$, from which the same result follows (†).

Likewise, in $S5$, $K_x(\neg K_x\alpha \vee \neg K_x\beta)$ is equivalent to $(\neg K_x\alpha \vee \neg K_x\beta)$ (for '$\Rightarrow$', use veridicality, for '$\Leftarrow$', observe that $\neg K_x\alpha$ implies $K_x\neg K_x\alpha$ which in turn implies $K_x(\neg K_x\alpha \vee \neg K_x\beta)$). Likewise, $\neg K_x\beta$ implies $K_x(\neg K_x\alpha \vee \neg K_x\beta)$, from which the same result follows (††).

Let $\mathsf{Ag} = \{a_1, \ldots, a_n\}$. Let $x = a_i$, $y = a_j$ and $z = a_k$. Let $\alpha_r = (Kw_xA_r \nabla Kw_yA_r)$. Now,

$$
\begin{array}{lll}
G, M & \models K_x \bigvee\limits_{z \in \mathsf{Ag}} K_x\hat{K}_x(Kw_xZ \nabla Kw_yZ) & \\
\iff & G, M \models K_x(K_x\hat{K}_x\alpha_1 \vee \cdots \vee K_x\hat{K}_x\alpha_n) & \\
\iff & G, M \models (K_x\hat{K}_x\alpha_1 \vee \cdots \vee K_x\hat{K}_x\alpha_n) & \text{(from (†))} \\
\iff & G, M \models (\hat{K}_x\alpha_1 \vee \cdots \vee \hat{K}_x\alpha_n) & \\
 & \quad (\text{`} \Rightarrow \text{'} \text{ by veridicality, `} \Leftarrow \text{'} \text{ by negative introspection}) & \\
\iff & G, M \models K_x(\hat{K}_x\alpha_1 \vee \cdots \vee \hat{K}_x\alpha_n) & \text{(from (††))} \\
\iff & G, M \models K_x\hat{K}_x(\alpha_1 \vee \cdots \vee \alpha_n) & \text{(semantics of `}\hat{K}_x\text{' operator)} \\
\iff & G, M \models K_x\hat{K}_x(\bigvee\limits_{z \in \mathsf{Ag}} \alpha_z) & \\
\iff & G, M \models K_x\hat{K}_x(\bigvee\limits_{z \in \mathsf{Ag}} Kw_xZ \nabla Kw_yZ) &
\end{array}
$$

From Claim 1 through 7, and from Proposition 3.26, we conclude that: $\Sigma(\Pi_1) \subsetneq \Sigma(\Pi_3) \subsetneq \Sigma(\Pi_2) \subsetneq \Sigma(\Pi_5) = \Sigma(\Pi_4)$. □

*Proof of Proposition 3.27(2).* To prove that $\Sigma(\Pi_0) \not\subseteq \Sigma(\Pi_1)$, we show that no execution sequence of $\Pi_0$ is an execution sequence of $\Pi_1$. In $\Pi_0$ some designated agent will call some other agent more than once in an execution sequence, but in $\Pi_1$ no agent can call any other agent more than once in an execution sequence: after the first call between a pair of agents, they learn each other's secret and as such the epistemic calling condition for $\Pi_1$ will not be satisfied for a second call between both same agents, in the same execution sequence.

We show that $\Sigma(\Pi_0) \subsetneq \Sigma(\Pi_3)$ is the case by arguing for the following two claims.

**Claim 1: Every execution sequence of $\Pi_0$ is an execution sequence of $\Pi_3$.**

To prove this claim, let the four designated agents for $\Pi_0$ be $a_1, a_2, a_3, a_4$, and let us consider $\Pi_0$ in three stages, as follows.

The *first stage* of $\Pi_0$ is when one of the designated agents, say (without loss of generality) agent $a_1$, calls each of the other $n-4$ non-designated agents. Considering what the agents know, agent $a_1$ knows that it will learn the unique secret of each of the non-designated agents by calling each of them in those $n-4$ calls. Therefore, the epistemic calling condition for $\Pi_3$ is obviously satisfied for each of those calls.

In the *second stage*, first, any two of the designated agents call each other; secondly, the other two agents call each other; thirdly, one of the first pair of callers in this stage calls one of the second pair of callers of the stage; fourthly, the two agents who were not in the third call of this stage call each other. Again, considering what the agents know in each of the calls in this second stage, notice that each calling pair in all four calls call each other for the first time. So likewise

they each know they will learn each other's unique secret. And, as in the first stage, the epistemic calling condition for $\Pi_3$ is also satisfied in each call in this stage.

Finally, in the *third stage*, agent $a_1$ calls each of all the $n-4$ non-designated agents once more. Again, from the point of view of what the agents know, let us consider agent $a_1$'s knowledge after the second stage of $\Pi_0$. After the last call of the second stage, agent $a_1$ does not consider it possible that there is a gossip situation in which it does not yet know all the secrets in the scenario. Also, after the last call of the second stage, agent $a_1$ does not consider it possible that there is a gossip situation in which any of the non-designated agents know all the secrets. In other words, agent $a_1$ knows that it knows all the secrets in the scenario, and agent $a_1$ knows that none of the non-designated agents know all the secrets in the scenario. Furthermore, since none of the non-designated agents took part in any call of the second stage, and since the other designated agents, apart from agent $a_1$, made all their calls in the second stage, it follows that none of the non-designated agents know the unique secret of any of the designated agents apart from agent $a_1$, in any gossip situation in the scenario. It follows therefore that agent $a_1$ does not consider it possible that there is a gossip situation in which any of the non-designated agents knows the unique secret of any of the other designated agents. In other words, for each non-designated agent $x$, and for each of the other designated agents, $y$, agent $a_1$ knows that agent $x$ does not know the secret of agent $y$. Since all the calls in the third stage of $\Pi_0$ involves agent $a_1$ calling each of the non-designated agents once, (let the non-designated agent whom agent $a_1$ calls in any call of the third stage be denoted by $x$) it follows that for each of those calls in the third stage, agent $a_1$ knows of a secret, namely that of another designated agent $y$, which agent $x$ will learn from the third stage call. Thus the epistemic calling condition for $\Pi_3$ is satisfied for each of the calls in the third stage.

We therefore conclude from the foregoing that an execution sequence of $\Pi_0$ is also an execution sequence of $\Pi_3$.

**Claim 2: Not every execution sequence of $\Pi_2$ is an execution sequence of $\Pi_0$.** To prove this claim, we show that there exists an execution sequence of protocol $\Pi_2$ that is not an execution sequence of $\Pi_0$. For this, consider a gossip scenario with four agents $a_1, a_2, a_3, a_4$. Now, the following execution sequence $\sigma$ is an example execution sequence of $\Pi_2$: $\sigma = a_1a_2; a_1a_3; a_1a_4; a_2a_3; a_2a_4; a_3a_4$. In each call in $\sigma$ each pair of agents call each other for the first time, so the calling agent knows that it will learn the unique secret of the agent it calls in each call. On the other hand, it is obvious that $\sigma$ is not an execution sequence of $\Pi_0$: consider, for example, that for $\Pi_0$, an execution sequence cannot be longer than four calls for a gossip scenario with exactly four agents.

Finally, the proof of the proposition that $\Sigma(\Pi_0) \subsetneq \Sigma(\Pi_3)$ follows from the proposition that $\Sigma(\Pi_3) \subsetneq \Sigma(\Pi_2)$ (see Proposition 3.27(1)) and from the foregoing proposition that

$\Sigma(\Pi_0) \subsetneq \Sigma(\Pi_2)$. And likewise, the proof of the proposition that $\Sigma(\Pi_0) \subsetneq \Sigma(\Pi_5)$ follows from the proposition that $\Sigma(\Pi_5) = \Sigma(\Pi_4)$ (see also Proposition 3.27(1)) and from the foregoing proposition that $\Sigma(\Pi_0) \subsetneq \Sigma(\Pi_4)$. □

**Proposition 3.28.** *Let* $\Pi =^s \Pi'$ *denote that the shortest sequence in* $\Sigma(\Pi)$ *has the same length as the shortest sequence in* $\Sigma(\Pi')$. *Then:*

$$\Pi_0 =^s \Pi_1 =^s \Pi_2 =^s \Pi_4$$

*Proof.* To prove this claim we observe that $\Pi_1 =^s \Pi_2 =^s \Pi_4$ follows from Proposition 3.27(1). Whereas $\Pi_0 =^s \Pi_4$ follows from Proposition 3.27(2). Therefore we conclude that:

$$\Pi_0 =^s \Pi_1 =^s \Pi_2 =^s \Pi_4$$ □

**Proposition 3.29.** *Let the average execution length* $EL(\Pi)$ *be the average length of* $\sigma \in \Sigma(\Pi)$ *if this set is finite, and* $\infty$ *otherwise. Let* $\Pi <^e \Pi'$ *denote that either* $EL(\Pi) < EL(\Pi') \in \mathbb{N}$, *or* $EL(\Pi) \neq \infty = EL(\Pi')$. *Then,*

$$\Pi_0 <^e \Pi_1 \text{ and } \Pi_2 <^e \Pi_4$$

*Proof.* Recall that the length of any execution sequence of $\Pi_0$ is $2n-4$, where $n = |Ag|$. To prove that $\Pi_0 <^e \Pi_1$, it is sufficient to show that there is an execution sequence $\Pi_1$ for $n$ agents that is greater than $2n-4$.
So let $Ag = \{a_1, \ldots, a_n\}$. Now consider the following execution sequence:

$$\sigma = a_1a_2; a_1a_3; \ldots; a_1a_n; a_2a_3; \ldots; a_2a_n; a_3a_4; \ldots; a_{n-1}a_n$$

The execution sequence $\sigma$ describes an instance where we select each agent in turn to call all the other agents whom it does not yet know their unique secret. Obviously this gives rise to an execution sequence $\sigma$ of $\Pi_1$. Furthermore, the length of this execution sequence $\sigma$ is $n(n-1)/2$. Thus we conclude that: $\Pi_0 <^e \Pi_1$.

To show that $\Pi_2 <^e \Pi_4$, we prove that whereas all the execution sequences of $\Pi_2$ are finite, some execution sequences of $\Pi_4$ are infinite.

Now, notice that $\Pi_5$ (and, therefore $\Pi_4$, see Proposition 3.27) may loop and therefore termination is not guaranteed. This means that some sequences of $\Pi_5$ (and $\Pi_4$) are infinite. For example, in case of a scenario with four agents, consider the following infinite sequence $\sigma \in \Sigma(\Pi_5)$: $\sigma = ab; cd; ab; cd; ab; \ldots$. After every even round (that is, after every call $cd$) in $\sigma$, we have $K_a\hat{K}_a(Kw_aC \nabla Kw_bC)$, that is, agent $a$ considers it possible that agent $b$ has learned secret $C$, while secret $C$ is unknown to agent $a$, namely if the second call were $bc$. Therefore, after the call sequence $ab; cd$, call $ab$ can be made, according to the protocol.

But unlike $\Pi_4$, both $\Pi_2$ and $\Pi_3$ terminate, as is argued as follows. Consider the set $S = \{(a, B) \mid \neg Kw_aB\}$. Initially, $|S| = n(n-1)$. The epistemic calling condition for the

Known Information Growth protocols implies that $S \neq \emptyset$, and, moreover, every round of the protocol removes at least one member from $S$. □

## 3.7 Conclusion

In this chapter, we proposed epistemic gossip protocols, where an agent will call another agent based on its current knowledge. We described various such protocols, we gave some of their logical properties, and we modelled them in dynamic epistemic logic.

We considered the case where only one pairwise call is staged in a given round of gossiping. There are other alternatives however, namely, *parallel calls* in which more than one pairwise call can be staged simultaneously. One strategy for allowing the execution of parallel calls in an epistemic gossip protocols could be to allow each of all the agents to *try* to make a call with one of the agents with whom the calling condition is satisfied for the given protocol and for the given network topology. Another strategy for making parallel calls could be through $k$-party calls, in which an agent makes a *conference call* with all the agents with whom it can call in a given situation. For example, if the calling condition is satisfied for an agent $a$ to call agent $b$ and agent $c$, then we can set up a conference call among agents $a, b$ and $c$ in which all three agents exchange their secrets among themselves.

Furthermore, in this thesis we assumed that in each pairwise call the calling pair exchange all the secrets they know. This assumption can be relaxed so that one can also consider *one-way calls*, that is, calls in which only one of the callers sends its secrets to the other calling partner. This case is analogous to *text messaging* or *electronic mail* between the communicating pair. And to take this line still further, one can consider a case where one of the agents broadcasts its secrets to a group of agents rather than to one other agent, similar to the popular manner in which information in shared in social networks among a group of friends. Calls such as one-way and broadcast calls can be used to describe the spread of a disease, a news item, or a commercial within a population.

Epistemic gossip protocols that are based on parallel calls, one-way calls, broadcast calls, or any combination of various such types of calls could also be described and analysed.

Another interesting line of future work is to consider strategic issues. Suppose the agents are allowed to choose from a set of protocols, or from a set of possible calls due to a protocol, can an agent ensure, for example, that it is the first to know all secrets, or, for that matter not the last?

# Chapter 4

# A Framework for Epistemic Gossip Protocols

## 4.1 Introduction

In this chapter we present Epistemic Gossip Protocol (EGP), which is a tool to analyse epistemic gossip protocols. Particularly we introduce Epistemic Gossip Protocol Language (EGPL), which is a high-level programming language for epistemic gossip protocols. Then, we describe the details of an interpreter for the EGPL. The tool EGP outputs key dynamic properties of an epistemic gossip protocol. In the next chapter we apply this tool to the epistemic gossip protocols introduced in Chapter 3, and then, we present empirical results.

The initial setting of the gossip scenarios we consider is as follows. There are a finite number of agents, and each agent knows a unique piece of information called a secret. Only pairwise communications between the agents are allowed. These communications are known as *calls*, and only one call is allowed in a round. In each call, the calling pair exchange all the secrets they know. The goal of such communications is to reach a state where all the agents know all the secrets in the scenario.

Each epistemic gossip protocol can be considered as a rule with some epistemic condition which has to be satisfied for one agent to call another agent. We call such rule a *calling condition*[*]. In each round, a pair of agents is chosen non-deterministically from the set of pairs for which the calling condition is satisfied, and allowed to make a call. This call can be made in one of several modes. For example, the calling pair can make the call publicly such that every other agent knows who is calling who in any round. This mode is referred to as the *public synchronous* mode (while there is no uncertainty

[*]Note that a calling condition is not limited only to an epistemic (calling) condition (epistemic calling condition is defined in Section 3.6), but can also include conditions about the underlying communication graph structure. For example, a pair of agents cannot engage each other in a pairwise call if they are not neighbours on the underlying network graph. Furthermore, the reader should assume a *complete* communication graph wheresoever in this thesis we have not made explicit the structure of the underlying communication graph. Note that our discussions so far in this thesis have assumed a complete graph as the underlying network graph, and as such the calling condition for a protocol have so far been tantamount to an epistemic calling condition.

about who is calling whom in each round, the contents of the calls, namely the secrets, are not observed). Another mode is that in which the calls are made in private such that, apart from the pair involved in the call, the other agents may not be sure which pair of agents is making the call, but all the agents are sure a call is made in each round. This mode is referred to as the *private synchronous* mode. The *private asynchronous* mode is like the private synchronous mode except that the agents consider it possible that no call is made in a round even though there is some pair for which the calling condition is satisfied.

In this chapter, we assume that the protocols are based on the private synchronous call mode. Therefore the agents would have to reason about possible situations which are due to all the possible calls in the previous round. For example: at the initial situation of a gossip scenario comprising of four agents, no other situation is considered possible. But after one round of calls, there could be up to twelve new and different possible situations due to possible calls at the initial situation. Note that we distinguish between the call $a_ia_j$ and the call $a_ja_i$, where $a_ia_j$ denotes the call from agent $a_i$ to another agent, $a_j$. Hence after a maximal series of rounds we can think of a tree structure in which each path is an execution sequence of calls of a given protocol. We refer to this tree as the *call tree* or *gossip tree* of the given protocol, and the set of all the paths of this tree is the *extension* of the protocol.

The gossip tree offers a platform to protocol designers for the evaluation and comparison of epistemic gossip protocols. In the gossip protocol literature it is typical to measure the performance of a protocol by considering the length of its execution sequence, that is, the number of calls in the execution sequence [30]. Correspondingly we measure the performance of epistemic gossip protocols by considering the average length of the execution sequences in the protocol's extension, together with the size of the extension of the given protocol. Whereas the size of the extension gives an idea of the computational memory required by an agent to reason about possible situations, the average execution length gives an idea of how fast it will take for all agents to know all secrets under the given protocol. We also make use of the following definitions.

**Definition 4.1** (Initial State and Goal State)**.** Given a set $\mathsf{Ag} = \{a_1, \ldots, a_n\}$ of agents and a set $P = \{A_1, \ldots, A_n\}$ of secrets. Let secret $A_i$ be the unique secret of $a_i$, and let $S_i$ be the set of secrets known by $a_i$ where $a_i \in \mathsf{Ag}$, and where initially $S_i = \{A_i\}$. Then, a *gossip situation* is a $n$-tuple $\langle S_1, \ldots, S_n\rangle$, the *initial state* is $\langle\{A_i\}, \ldots, \{A_n\}\rangle$, and the *goal state* is $\langle P, \ldots, P\rangle$.

**Definition 4.2** (Terminating and Non-terminating Execution Sequences)**.** An execution sequence is terminating if and only if it is finite. An execution sequence is non-terminating if and only if it is infinite.

**Definition 4.3** (Terminating and Non-terminating Protocol)**.** An epistemic gossip protocol is *terminating* if all its execution sequences are finite. Otherwise the protocol is *non-terminating*.

The following protocols were described in Chapter 3, reproduced informally here.

1. **Learn New Secrets.** An agent $a_i$ can call another agent $a_j$ if $a_i$ does not know the secret of $a_j$.

2. **Known Information Growth de Dicto.** An agent $a_i$ can call another agent $a_j$ if $a_i$ knows that there is *some secret* $A_k$ that would be learnt in the call $a_i a_j$.

3. **Known Information Growth de Re.** An agent $a_i$ can call another agent $a_j$ if there is *some secret* $A_k$ such that $a_i$ knows that it would be learnt in the call $a_i a_j$.

4. **Possible Information Growth de Dicto.** An agent $a_i$ can call another agent $a_j$ if $a_i$ considers it possible that there is *some secret* $A_k$ that would be learnt in the call $a_i a_j$.

5. **Possible Information Growth de Re.** An agent $a_i$ can call another agent $a_j$ if there is *some secret* $A_k$ such that $a_i$ considers it possible that it would be learnt in the call $a_i a_j$.

Whereas Protocols 1, 2 and 3 are terminating, Protocols 4 and 5 are non-terminating. Take a scenario with four agents $a, b, c, d$ and consider the following execution sequence of Protocol 4: $ab; cd; ab; cd; \ldots$. After the first two calls, agent $a$ considers it possible that agent $b$ learnt some new secret in the second round, therefore $a$ calls $b$ in the third round, which turns out to be redundant. Likewise in the fourth round agent $c$ considers it possible that agent $d$ learnt some new secret in the third round, so $c$ calls $d$ in the fourth round. In this way the loop $ab; cd; \ldots$ could go on infinitely. The same example works for Protocol 5.

In a given round of gossiping, an agent may be uncertain about the actual execution sequence that occurred. However the agent in question may consider some execution sequences possible based on the fact that it cannot distinguish between these execution sequences. Under Known Information Growth de Dicto and Known Information Growth de Re, for agent $a_i$ to call agent $a_j$ at a gossip situation, it is required that at every execution sequence that agent $a_i$ considers possible at the given gossip situation, there is some secret that only one of agent $a_i$ and $a_j$ knows. As such, if the call $a_i a_j$ is possible at the given gossip situation, then one of agent $a_i$ and $a_j$ is sure to learn some new secret from the other.

The difference between the Known Information Growth and Possible Information Growth protocols is as follows. Whereas in Known Information Growth de Re agent $a_i$ is certain of the particular secret that will be learnt in such $a_i a_j$ call, in Known Information Growth de Dicto $a_i$ may remain uncertain about the particular secret that will be learnt in the $a_i a_j$ call. For example, consider again the gossip scenario with four agents $a, b, c, d$. After the execution sequence $ab; bc; cd$, agent $a$ is certain that it will learn some new secret by calling agent $b$ in the fourth round (since agent $b$ must have been involved in some call in either the second round or the third round), however, agent

$a$ is uncertain about the secret it will learn from agent $b$ in the fourth round (since agent $b$ may have called with either agent $c$ or agent $d$ in the second or third round). Therefore the epistemic calling condition for Known Information Growth de Dicto holds for agent $a$ to call $b$ in the fourth round, but not for Known Information Growth de Re. But, for the same scenario and for the same execution sequence, in the fourth round, agent $a$ is certain that it will learn the secret of agent $c$ in an $ac$ call. Thus the epistemic calling condition for Known Information Growth de Re is satisfied. For a broader discussion of the De Re / De Dicto distinction, see [34]. The difference between Possible Information Growth de Dicto and Possible Information Growth de Re protocols is obvious from the foregoing discussion.

## 4.2 The EGP Tool

The epistemic gossip protocol extension and gossip tree are convenient for design and planning purposes. However they are not easy to construct manually, even for a small number of agents. The difficulty of such a task naturally increases with the number of agents and with the complexity of the epistemic property comprising the epistemic calling condition for the given protocol. Therefore it is desirable to have a tool that automates the process of gossip tree generation and the evaluation of epistemic gossip protocols by means of their extension. We implement such a tool, namely, the EGP tool. Given a high level description of an epistemic gossip protocol, the EGP tool outputs the characteristics of the epistemic gossip protocol by analysing the extension of the given protocol.

In this section we present the implementation structure of the EGP tool, describing each of its components. The EGP tool comprises of a high-level programming language, EGPL, an interpreter for EGPL and the EGPL Modeller. The EGPL Modeller accepts the calling condition for one agent $a$ to call another agent $b$, and checks whether this calling condition is satisfied at a given situation (or node) in the gossip tree. If this calling condition is satisfied at a situation, the EGPL Modeller extends the gossip tree by creating a new situation which is as a result of a call from agent $a$ to agent $b$ in the current situation. The checking of the calling condition at a situation is done by the Model Checker whereas the Gossip Tree Generator extends the gossip tree when new situations are created. The EGPL Modeller also carries out an analysis of a generated gossip tree and outputs protocol characteristics and sample execution sequences. See Figure 4.1 for the structural overview of the EGP tool.

**Definition 4.4.** The language $\mathcal{L}_{cc}$ is defined as

$$\mathcal{L}_{cc} \ni \quad \varphi \quad ::= \quad Kw_{a_i}A_j \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \rightarrow \varphi) \mid K_{a_i}\varphi$$

Note that $A_j$ is a propositional atom, and the formula $Kw_{a_i}A_j$ stands for 'agent $a_i$ knows whether secret $A_j$ (is true)', and $K_{a_i}\varphi$ stands for 'agent $a_i$ knows that $\varphi$ (is true)'.

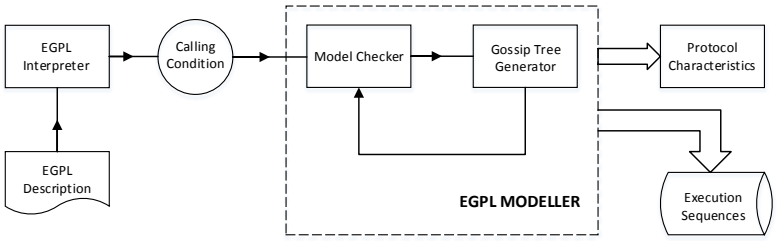



FIGURE 4.1: **Structural overview of the EGP tool.**

### 4.2.1 Structural Overview

For the protocol designer we provide a high level language for describing epistemic gossip protocols within the EGP tool. We call this language Epistemic Gossip Protocol Language (EGPL). Given the epistemic calling condition $\varphi_\Pi(a_i, a_j) \in \mathcal{L}_{cc}$ required for any agent $a_i$ to call another agent $a_j$ under an epistemic gossip protocol $\Pi$, the language EGPL is designed such that $\varphi_\Pi(a_i, a_j)$ can also be expressed in the language of EGPL (although EGPL as a programming language is capable of expressing other properties that are not expressible in the formal language $\mathcal{L}_{cc}$; we refer the reader to Appendix A for the BNF grammar, symbols and operators of the EGPL). In this thesis however, we limit the application of the EGP tool to those protocols whose epistemic calling condition is expressible in the language of $\mathcal{L}_{cc}$.

Next we present a language interpreter for EGPL. Given a protocol description expressed in EGPL, the EGPL interpreter generates the gossip tree corresponding to the described protocol, and outputs the characteristics of the protocol. The terminating protocols are characterised in terms of (a) *successfulness* (b) *average execution length* (c) *shortest and longest execution lengths* and (d) *extension size*. Also, samples of execution sequences of various lengths are displayed by the tool, whereas the entire extension of the protocol is stored in a file.

To generate the possible situations in a round, the EGPL interpreter employs an epistemic model checker to check the calling conditions for each pair of agents at each possible situation.

### 4.2.2 Epistemic Gossip Protocol Language (EGPL)

EGPL is a programming language for describing epistemic gossip protocols in terms of the epistemic calling conditions of such protocols. An EGPL protocol description starts with the keyword `begin` and finish with the keyword `end`. Between these two keywords lies the core of the protocol specification which consists of the calling condition of the protocol. The following code listing is an example protocol specification using EGPL. The protocol described is Protocol 1.

```
1  begin
2      /* epistemic calling condition */
3      let ai call aj if {
4         ai knows (init(aj) \notin secret(ai));
5      }
6  end
```

LISTING 4.1: **EGPL Description for Protocol** 1

In Listing 4.1, the epistemic calling condition is given in lines 3-5. It says that agent `ai` can call agent `aj` if `ai` knows that it does not know `aj`'s unique secret. In the EGPL description in Listing 4.1, `ai` and `aj` are agent name variables. They are substituted by agent names when the description is parsed.

### 4.2.3 The EGPL Interpreter

As shown in Figure 4.1, the framework tool accepts an EGPL protocol description, and outputs a set of protocol characteristics and execution sequences for the described protocol.

An EGPL description is interpreted in two stages, namely, expansion stage and model checking stage. The expansion stage produces an instance of the epistemic calling condition for every valid agent combination. A valid agent combination is obtained by substituting a unique and real agent name for each unique agent name variable appearing in the description. The output of the expansion stage is the set of all the calling condition instances, over all the agents in the scenario. See listing 4.2 for a sample output from the expansion stage for the description shown in listing 4.1. In this example the names for the agents in the scenario are $a, b, c, d$.

```
c knows((((init(b)\notinsecret(c)))));
c knows((((init(a)\notinsecret(c)))));
b knows((((init(a)\notinsecret(b)))));
d knows((((init(a)\notinsecret(d)))));
d knows((((init(c)\notinsecret(d)))));
d knows((((init(b)\notinsecret(d)))));
c knows((((init(d)\notinsecret(c)))));
b knows((((init(d)\notinsecret(b)))));
a knows((((init(b)\notinsecret(a)))));
a knows((((init(c)\notinsecret(a)))));
b knows((((init(c)\notinsecret(b)))));
a knows((((init(d)\notinsecret(a)))));
```

LISTING 4.2: **Expansion Stage Output (Four-Agent Scenario)**

The set of all the calling condition instances is fed into the model checking stage where they are checked on all the possible situations in the round to determine which pair of agents satisfy the epistemic calling condition at the considered situation.

The parsers are implementations of the Look-Ahead Left to Right (LALR) parsing technique described in [19]. The technique LALR usually refers to the more specific technique of LALR(1) parsing in which there is one token look ahead during parsing in

order to resolve the differences between language production rules. The Left to Right, LR(k), parsing technique was invented by Donald Knuth who also showed that any LR(k) grammar can be transformed into an LR(1) grammar, where $k$ is the number of lookahead terminals of language production rules and $k > 1$ [38]. Lookahead terminals allow the specification of richer languages out of a few simple rules, while providing other useful parsing utilities. However the use of lookahead terminals incurs a large memory overhead for the parser. Frank DeRemer introduced the LALR(1) technique as a more memory efficient alternative to LR(1) technique. Although the LALR(1) comes with a weaker language recognition power than the LR(1) technique, it still has sufficient power to parse the grammar of many useful and mainstream programming languages, for example Java. Finally, we use Constructor of Useful Parsers (CUP) parser generator to automate the construction of the LALR parser for the EGPL programming language. See [31] for details of CUP parser generator.

### 4.2.4 Gossip Tree Generator

The EGPL Modeller is the component that constructs and updates the gossip tree for the specified protocol. It consists of two main components, namely, a gossip tree generator and a model checker. The model we refer to is a partial gossip tree which is an abstraction of the gossip model described in Chapter 3 (see also [3]). Given the *initial* or *root node* of the gossip tree, the gossip tree generator constructs the successor nodes of the gossip tree for the specified protocol. Each successor node in the gossip tree is a result of a possible call at some *parent node*. Such a call is considered possible at the parent node if the calling condition given by the specified protocol is satisfied at the parent node. The model checker is needed to construct the gossip tree because it checks the calling conditions on the nodes of the gossip tree to determine successor nodes. In what follows in this section we describe the gossip tree and provide the semantics of the language of epistemic calling conditions based on the gossip tree. Let us begin as follows by defining some of the building blocks of the gossip tree.

**Definition 4.5** (History, $h$)**.** A history $h$, or an execution sequence, is defined inductively as: $h ::= e \mid h; a_i a_j$ where $e$ is the empty sequence and $a_i \neq a_j \in \mathsf{Ag}$.

**Definition 4.6** (Situation Label, $F$)**.** A *gossip situation* is a tuple $\theta = \langle S_1, \ldots, S_n \rangle$. We let $\Theta$ be the set of every $\theta$. Given a non-empty set $H$ of histories, a *situation label* for $H$ is a function $F : H \to \Theta$ that returns the gossip situation corresponding to a given history. The function $F$ is defined as $F(e) = \langle \{A_1\}, \ldots, \{A_n\} \rangle$ where $A_i$ is the secret of agent $a_i$, and if $F(h) = \langle S_1, \ldots, S_n \rangle$ then $F(h; a_i a_j) = \langle S'_1, \ldots, S'_n \rangle$, where $S'_i = S_i \cup S_j = S'_j$ and $S'_k = S_k, \quad k \notin \{i, j\}$.

**Definition 4.7** (Equivalence Relation, $\equiv$)**.** Let $F(h; a_i a_j) = \langle S_1, \ldots, S_n \rangle$, and $F(h'; a_k a_l) = \langle S'_1, \ldots, S'_n \rangle$. Let the call mode be 0-mode. We inductively define an equivalence relation between histories as follows:

**Base Case:** $e \equiv_{a_m} e$

**Inductive Case:** $h; a_i a_j \equiv_{a_m} h'; a_k a_l$ iff: $S_m = S'_m$, and $h \equiv_{a_m} h'$,
and $[a_m \in \{a_i, a_j\} = \{a_k, a_l\}$ or $a_m \notin \{a_i, a_j\} \cup \{a_k, a_l\}]$, for all $a_m \in \mathsf{Ag}$.

The definition of equivalence relation given in Definition 4.7 is for the private synchronous call mode, in which case it is common knowledge among all the agents that a call is made in each round of gossiping if and only if there is some pair of agents for which the calling condition of the protocol is satisfied. Moreover the other agents who were not involved in a call may be uncertain as to which pair of agents were involved in the call. According to the equivalence relation given in Definition 4.7, the history corresponding to the initial situation is the empty history. After the call $a_i a_j$, both $a_i$ and $a_j$ know the same secrets, whereas the secret known by other agents who were not in the call remains the same as it was before the $a_i a_j$ call. Two histories are equivalent for an agent $a_m$ if they are both of the same length; and, after each corresponding round for both histories, $a_m$ knows the same secrets in both histories; and, in each corresponding round for both histories, $a_m$ is either not involved in the call, or $a_m$ called with the same other agent in that round. To give an example, consider a gossip scenario with four agents: $a, b, c, d$. Let $\sigma_1 = ab; dc; ac; bd$ and $\sigma_2 = ab; ac; ad; db$, then according to Definition 4.7, $\sigma_1 \equiv_b \sigma_2$. We refer the reader to Section 4.5 for a further discussion about equivalence of histories.

**Definition 4.8** (Tree)**.** A tree is a tuple $\mathcal{T} = \langle H, R \rangle$, where $H$ is a finite set of histories closed under prefixes, and $R$ is a *parent relation* over $H$ such that $h' R h$ if there exists $a_i, a_j$ such that $h = h'; a_i a_j$. We call $h'$ the *parent node* of such $h$. The node $e$ is the root node of the tree.

**Definition 4.9** (Epistemic Tree)**.** An *epistemic tree* is a quadruple $\langle H, R, F, \{Z_{a_m}\} \rangle$ where $\langle H, R \rangle$ is a tree, $F$ is a situation label for $H$, and $Z_{a_m} : H \to 2^H$ is a function that assigns an equivalence class to agent $a_m \in \mathsf{Ag}$ from the domain of histories such that $h \equiv_{a_m} h'$ for all $h' \in Z_{a_m}(h)$. Furthermore, we define a parent relation $R_c$ over the codomain of all $Z_{a_m}$ such that $Z_{a_m}(h'') R_c\, Z_{a_m}(h)$ if and only if for every node $h' \in Z_{a_m}(h)$, the parent node of $h'$ is in $Z_{a_m}(h'')$.

Note that where $\mathcal{T} = \langle H, R, F, \{Z_{a_m}\} \rangle$, then for $h \in H$ we may also write $h \in \mathcal{T}_g$. We will sometimes refer to such equivalence class $Z_{a_m}(h)$ of the node $h$ as the *cell* of $h$. If $Z_{a_m}(h'') R_c\, Z_{a_m}(h)$, then $Z_{a_m}(h'')$ is called the *parent cell* of $Z_{a_m}(h)$, and $Z_{a_m}(h)$ is called the *child cell* of $Z_{a_m}(h'')$.

**Definition 4.10** (Epistemic Tree Layer)**.** Given an epistemic tree $\mathcal{T}$, let $k = l(h)$ be the length of $h \in H$; let $l(e) = 0$ and $l(h; a_i a_j) = l(h) + 1$. Then the *layers* $\lambda$ of $\mathcal{T}$ is defined as a tuple $\lambda = \langle \lambda_0, \lambda_1, \dots \rangle$ where $\lambda_k = \{h \mid h \in H \text{ and } l(h) = k\}$.

For any $\lambda_k$ of an epistemic tree $\mathcal{T}$, and for any $h \in \lambda_k$, the equivalence class of $h$ for all agents, the parent relations involving $h$, and the situation label of $h$ are the same as in $\mathcal{T}$. Finally, we define a labelling function $L$ over $\lambda$ such that $L_{a_m}(\lambda_k) = \bigcup_{h \in \lambda_k} \{Z_{a_m}(h)\}$, for all $a_m \in \mathsf{Ag}$.

The labelling function $L$ defines a partition, for each agent, over the nodes in a given layer. Hence $Z_{a_i}(h)$ returns the cell of $h$ in the partition corresponding to agent $a_i$ in a given epistemic tree layer.

**Definition 4.11** (Interpreting Formulas of $\mathcal{L}_{cc}$ on epistemic trees)**.** Given a layer $\lambda_k$ of any epistemic tree $\mathcal{T}$, we inductively define the interpretation of a formula $\varphi \in \mathcal{L}_{cc}$ on a node $h \in \lambda_k$ as follows[†]:

$$
\begin{array}{lll}
\lambda_k, h \models Kw_{a_i}A_j & \text{iff} & A_j \in S_i, \text{ where } S_i \text{ is the } i^{th} \text{ item in } F(h) \\
\lambda_k, h \models \neg\varphi & \text{iff} & \lambda_k, h \not\models \varphi \\
\lambda_k, h \models (\varphi \wedge \psi) & \text{iff} & \lambda_k, h \models \varphi \quad \text{and} \quad \lambda_k, h \models \psi \\
\lambda_k, h \models (\varphi \vee \psi) & \text{iff} & \lambda_k, h \models \varphi \quad \text{or} \quad \lambda_k, h \models \psi \\
\lambda_k, h \models (\varphi \rightarrow \psi) & \text{iff} & \lambda_k, h \models \neg\varphi \quad \text{or} \quad \lambda_k, h \models \psi \\
\lambda_k, h \models K_{a_i}\varphi & \text{iff} & \lambda_k, h' \models \varphi \quad \text{for every } h' \in Z_{a_i}(h) \\
\lambda_k, h \models \varphi & \text{iff} & \mathcal{T}, \lambda_k, h \models \varphi \quad \text{iff} \quad \mathcal{T}, h \models \varphi \\
\lambda_k \models \varphi & \text{iff} & \lambda_k, h \models \varphi \quad \text{for all } h \in \lambda_k \\
\mathcal{T} \models \varphi & \text{iff} & \mathcal{T}, h \models \varphi \quad \text{for all } h \in \mathcal{T}
\end{array}
$$

Before we go on to the definition of an execution sequence of a protocol, we will first give an ancillary definition as follows.

**Definition 4.12** (Epistemic Context of an Execution Sequence)**.** Given an epistemic gossip protocol $\Pi$ and given any history $h$, let the epistemic calling condition for any agent $a$ to call another agent $b$ under protocol $\Pi$ be $\varphi_\Pi(a, b)$. Then the *epistemic context* $\mathcal{C}$ of $h$, written $\mathcal{C}(h)$, is an epistemic tree $\mathcal{T}^h = \langle H^h, R, F, \{Z_{a_m}\}\rangle$, where $R, F$ and $Z_{a_m}$ are as given in Definition 4.9, and $H^h$ is defined inductively as follows:

- If $h = e$ then $H^h = \{e\}$
- If $h = h'; a_i a_j$ then $H^h = \{h''; ab \mid h'' \in H^{h'},\ a, b \in \mathsf{Ag} \text{ and } \mathcal{C}(h'), h'' \models \varphi_\Pi(a, b)\}$

Where $\mathcal{C}(h)$ is the epistemic tree $\mathcal{T}^h = \langle H^h, R, F, \{Z_{a_m}\}\rangle$, we will sometimes also abuse notation and write $h' \in \mathcal{C}(h)$ for $h' \in H^h$, and we will sometimes say that such $h'$ is in the epistemic context of $h$.

**Definition 4.13** (Execution Sequence of an Epistemic Gossip Protocol)**.** Given an epistemic gossip protocol $\Pi$ and given any history $h$. Let $\varphi_\Pi(a_i, a_j)$ be the epistemic calling condition for some agent $a_i$ to call another agent $a_j$ under protocol $\Pi$. Then $h$ is an execution sequence of $\Pi$ if and only if one of the following conditions hold:

- $h = e$
- $h = h'; a_i a_j$, and $h' \in H^{h'}$, and $\mathcal{C}(h'), h' \models \varphi_\Pi(a_i, a_j)$

[†]Note that the truth value of $A_j$ is irrelevant here, what is important is whether the truth value is known.

**Proposition 4.14.** *Given an epistemic gossip protocol* $\Pi$*, and a history* $h$*. Then* $h$ *is an execution sequence of* $\Pi$ *if and only if* $h$ *is in the epistemic context of* $h$*.*

*Proof.* Let the epistemic calling condition for any agent $a$ to call another agent $b$ under protocol $\Pi$ be $\varphi_\Pi(a_i, a_j)$. We proceed by induction over $h$, and distinguish the following cases.

**Base Case**

Suppose $h = e$, then from Definitions 4.12 and 4.13 we see that $e$ is an execution sequence of $\Pi$ if and only if $e \in \mathcal{C}(e)$.

**Inductive Case**

Suppose that $h = h'; a_i a_j$, and suppose that $h'$ is an execution sequence of $\Pi$ if and only if $h'$ is in the epistemic context of $h'$.

For the '$\Longrightarrow$'-direction, suppose that $h'; a_i a_j$ is an execution sequence of $\Pi$. Then from Definition 4.13 it follows that $\mathcal{C}(h'), h' \models \varphi_\Pi(a_i, a_j)$. But given the inductive hypothesis, this also implies that $h'; a_i a_j \in H^{h'; a_i a_j}$ as given in Definition 4.12, which is the same as saying that $h'; a_i a_j$ is in the epistemic context of $h$.

For the '$\Longleftarrow$'-direction, suppose that $h'; a_i a_j$ is in the epistemic context of $h$. Then from Definition 4.12 this implies that $h'; a_i a_j \in H^{h'; a_i a_j}$ and, given the inductive hypothesis, this implies that $\mathcal{C}(h'), h' \models \varphi_\Pi(a_i, a_j)$ holds, thus satisfying the conditions for $h'; a_i a_j$ to be an execution sequence of $\Pi$, from Definition 4.13. □

**Proposition 4.15.** *Given an epistemic gossip protocol* $\Pi$*, and any two histories* $h$ *and* $\hat{h}$*. If* $\hat{h}$ *is in the epistemic context of* $h$*, then* $\hat{h}$ *is an execution sequence of* $\Pi$*.*

*Proof.* Let the epistemic calling condition for any agent $a$ to call another agent $b$ under protocol $\Pi$ be $\varphi_\Pi(a_i, a_j)$. We proceed by induction over $\hat{h}$, and distinguish the following cases.

**Base Case**

Suppose $\hat{h} = e$, then from Definitions 4.12 and 4.13 we see that $e$ is an execution sequence of $\Pi$ if and only if $e \in \mathcal{C}(e)$. Therefore we conclude that if $\hat{h}$ is in the epistemic context of $h$, then $\hat{h}$ is an execution sequence of $\Pi$.

**Inductive Hypothesis**

Suppose that $\hat{h} = \hat{h}'; a_i a_j$, and that $h = h'; a_i a_j$. As the inductive hypothesis, suppose that it is the case that if $\hat{h}'$ is in the epistemic context of $h'$ then $\hat{h}'$ is an execution sequence of $\Pi$. Now assume that $\hat{h}'$ is in the epistemic context of $h'$, so we conclude that $\hat{h}'$ is an execution sequence of $\Pi$. Furthermore, suppose $\hat{h}'; a_i a_j$ is in the epistemic context of $h$. This implies, from Definition 4.12, that $\hat{h}' \in H^{h'}$, and that $\mathcal{C}(h'), \hat{h}' \models \varphi_\Pi(a_i, a_j)$ holds, thus satisfying the conditions for $\hat{h}'; a_i a_j$ to be an execution sequence of $\Pi$, from Definition 4.13. We therefore conclude that such a history $\hat{h}'; a_i a_j$ is an execution sequence of $\Pi$. □

A history $h$ may not be in its own epistemic context: Proposition 4.14 tells us the condition under which a history is in its own epistemic context, namely, if the history is an execution sequence of the given protocol. Now suppose that we are given two histories $h$ and $\hat{h}$, and an epistemic gossip protocol $\Pi$; let $h$ be an execution sequence of $\Pi$ while $\hat{h}$ is not an execution sequence of $\Pi$. Then we see that although $h$ may be in the epistemic context of $\hat{h}$, it is not the case that $\hat{h}$ is in the epistemic context of $h$. We see this from the contraposition of Proposition 4.15.

**Definition 4.16** (Gossip Tree, $\mathcal{T}_g$)**.** Given an epistemic gossip protocol $\Pi$, an epistemic tree $\langle H, R, F, \{Z_{a_m}\}\rangle$ is complete with respect to a protocol $\Pi$ if $H \subseteq \Sigma(\Pi)$, where $\Sigma(\Pi)$ is the extension of $\Pi$. An epistemic tree is *compliant* with $\Pi$ if $\Sigma(\Pi) \subseteq H$. An epistemic tree is a *gossip tree* for $\Pi$ if it is *compliant* with $\Pi$ and *complete* with respect to $\Pi$.

Note that if $\mathcal{T}_g$ is the gossip tree for a protocol $\Pi$, then such $h \in \mathcal{T}_g$ is an execution sequence of the protocol $\Pi$.

To illustrate a gossip tree, in Figure 4.2 we show the first three layers of the gossip tree for the Learn New Secrets protocol. In Figure 4.2 we use names $a, b, c, d$ for the agents in the scenario, with corresponding secrets $A, B, C, D$. In each layer shown we omit reverse calls for the sake of visual clarity.

**Definition 4.17** (Terminal Execution Sequence or History)**.** Given an epistemic gossip protocol $\Pi$, let $\mathcal{T}_g$ be the gossip tree for $\Pi$. Let $\varphi_\Pi(a_i, a_j)$ be the epistemic calling condition for $a_i$ to call $a_j$, where $a_i, a_j \in \mathsf{Ag}$. Then, an execution sequence or history $h \in \mathcal{T}_g$ is terminal if and only if for every pair $a_i, a_j \in \mathsf{Ag}$, $\quad \mathcal{T}_g, h \models \neg\varphi_\Pi(a_i, a_j)$.

Informally, an execution sequence or history $h$ is terminal if and only if no pair of agents can make any further calls at $h$, under the given protocol.

**Definition 4.18** (Successful and Unsuccessful Execution Sequence)**.** An execution sequence is *successful* if it is finite, and the first gossip situation is the initial state, and the last gossip situation is the goal state. An execution sequence is *unsuccessful* if it is *not* successful.

**Definition 4.19** (Successful and Unsuccessful Execution Sequence of an Epistemic Gossip Protocol)**.** Given an epistemic gossip protocol $\Pi$, let $\mathcal{T}_g$ be the gossip tree for $\Pi$. An execution sequence or history $h$ of $\Pi$ is successful if and only if: $h$ is terminal, and for every $a_i \in \mathsf{Ag}$, $\mathcal{T}_g, h \models \bigwedge_{a_k \in \mathsf{Ag}} Kw_{a_i} A_k$.

Informally, an execution sequence or history $h$ of a protocol is successful if at the gossip situation due to $h$, it is the case that every agent knows every secret, and no more calls are possible after $h$ under the given protocol. Notice that a successful execution sequence gives rise to the goal state.

**Definition 4.20** (Successful and Unsuccessful Protocol)**.** An epistemic gossip protocol $\Pi$ is successful if and only if: all the execution sequences or histories of $\Pi$ are terminating;

and, for every terminal execution sequence $h$ of $\Pi$, and for every $a_i \in \mathsf{Ag}$, $\mathcal{T}_g, h \models \bigwedge_{a_k \in \mathsf{Ag}} Kw_{a_i} A_k$.

**Definition 4.21** (Standard Extension of an Epistemic Gossip Protocol)**.** The *standard extension* of an epistemic gossip protocol is the set of all its terminal execution sequences of calls.

Notice that for a terminating epistemic gossip protocol $\Pi$, every execution sequence in the extension of $\Pi$ is also a prefix of some execution sequence in the standard extension of $\Pi$, whereas the standard extension of $\Pi$ is a strict subset of the extension of $\Pi$. By this property the standard extension serves an optimisation purpose, when we use it as a standard to compare terminating epistemic gossip protocols for *efficiency*, later in Chapter 5.

Note that in defining the gossip tree for a protocol $\Pi$, we made the following implicit assumptions:

1. the same protocol is followed by all the agents
2. the protocol is common knowledge among all the agents in the scenario
3. the agents remember the calls they were involved in within an execution sequence.

In the itemised assumptions above, Item 3 emerges from the definition of equivalence relation given in Definition 4.7. This notion of equivalence contrasts with that given in Definition 3.18, since in Definition 3.18 it is required only that an agent knows the same set of secrets in a pair of gossip situations in order to consider such pair equivalent. The difference between the two equivalence notions can be explained in terms of capabilities of agents: in Definition 4.7 the agents are enabled to keep track of whom they called with previously in an execution sequence, but in Definition 3.18 this capability is lacking in the agents whereupon they only distinguish gossip situations based on local information in the situation, namely, the set of secrets known, as against both the set of secrets known *and* history of calls, as required by Definition 4.7.

Let us illustrate the difference between the two equivalence notions using an example. In a gossip scenario with four agents $a, b, c, d$, consider the following two call sequences or histories:

$$\begin{aligned} h_1 &= ab; ac; bd \\ h_2 &= ac; bc; bd \end{aligned}$$

Let $(G', M')$ be the gossip situation resulting from executing $h_1$ at the initial gossip situation, and let $(G'', M'')$ be the gossip situation resulting from executing $h_2$ at the initial gossip situation. We then see that based on the equivalence notion in Definition 3.18, $(G', M')$ and $(G'', M'')$ are equivalent for agent $c$, since agent $c$ knows the same set of secrets in both situations, namely, {A, B, C}. However, based on Definition 4.7, $h_1$ and $h_2$ are clearly distinguishable for agent $c$, since, for example, agent $c$ knows that it

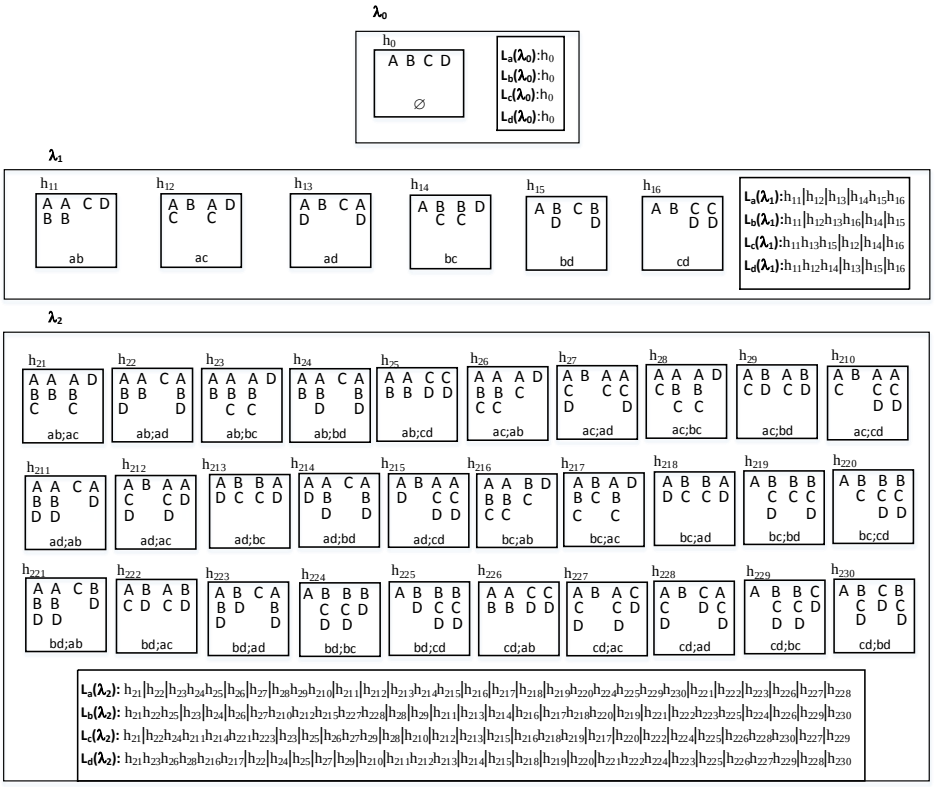



FIGURE 4.2: **Gossip tree layers (four-agent scenario).**

was involved in the first call of $h_2$ but not in the first call of $h_1$. Another example that shows that agent $c$ can distinguish between $(G', M')$ and $(G'', M'')$ based on Definition 4.7, is as follows: at $(G'', M'')$ agent $c$ knows that both agents $a$ and $b$ know secret $C$, but at $(G', M')$ agent $c$ does not know whether both agents $a$ and $b$ know secret $C$ as it considers it possible (in the private synchronous call mode) that the third call of $h_1$ may have been between agent $a$ and agent $b$ in which case agent $b$ too would have learnt secret $C$. But we see that without giving the agents the capability to remember calls they have made, agent $c$ may not know which agent knows secret $C$ in both execution sequences.

Although from this chapter onwards, we adopt the equivalence notion given in Definition 4.7, we would like to pause briefly here to treat a claim that if we adopt the same equivalence notion for both the gossip model and the epistemic tree, namely, that equivalence notion which is based only on what secrets the agents know as given in Definition 3.18, then the same properties are true on both the gossip model and the epistemic tree (see Proposition 4.23). Recall that throughout this chapter we assume that the call mode is the private synchronous mode.

**Definition 4.22.** Given a set $\mathsf{Ag}$ of $n$ agents, let $\mathcal{T}^* = \langle H^*, R, F, \{Z^*_{am}\}\rangle$ be a type of epistemic tree where:

- $H^*$ is the set of all histories
- $R$ is as in Definition 4.8
- $F$ is as in Definition 4.6
- $Z^*_{a_m} : H^* \to 2^{H^*}$ is a function that assigns an equivalence class to $a_m \in \mathsf{Ag}$ from the domain of histories such that: $\equiv^*_{a_m}$ is an equivalence relation, and $h \equiv^*_{a_m} h'$ if and only if $S_m = S'_m$, where $F(h) = \langle S_1, \ldots, S_n \rangle$, and $F(h') = \langle S'_1, \ldots, S'_n \rangle$, and $h, h' \in H^*$.
- The interpretation of a formula $\varphi \in \mathcal{L}_{cc}$ on a node $h \in \mathcal{T}^*$ is analogous to that in Definition 4.11, and given as follows:

$$\begin{array}{lll}
\mathcal{T}^*, h \models Kw_{a_i}A_j & \text{iff} & A_j \in S_i, \text{ where } S_i \text{ is the } i^{th} \text{ item in } F(h) \\
\mathcal{T}^*, h \models \neg\varphi & \text{iff} & \mathcal{T}^*, h \not\models \varphi \\
\mathcal{T}^*, h \models (\varphi \wedge \psi) & \text{iff} & \mathcal{T}^*, h \models \varphi \text{ and } \mathcal{T}^*, h \models \psi \\
\mathcal{T}^*, h \models (\varphi \vee \psi) & \text{iff} & \mathcal{T}^*, h \models \varphi \text{ or } \mathcal{T}^*, h \models \psi \\
\mathcal{T}^*, h \models (\varphi \to \psi) & \text{iff} & \mathcal{T}^*, h \models \neg\varphi \text{ or } \mathcal{T}^*, h \models \psi \\
\mathcal{T}^*, h \models K_{a_i}\varphi & \text{iff} & \mathcal{T}^*, h' \models \varphi \text{ for every } h' \in Z^*_{a_i}(h)
\end{array}$$

**Proposition 4.23.** *Let $\mathcal{T}^*$ be as given in Definition 4.22, and let the interpretation of a formula $\varphi \in \mathcal{L}_{cc}$ on a node $h \in \mathcal{T}^*$ be as given in Definition 4.22. Let $G = (\mathcal{S}, \approx)$ be a gossip model (see Definition 3.7 in Chapter 3), where $M_1$ is the initial gossip situation, and $\mathcal{S} = \{M_1\}$. Let $(G, M_1)[\![h]\!](G', M')$, where $[\![h]\!]$ is as defined in Definition 3.18 and Definition 3.19. Then*[‡]*:*

$$G', M' \models_g \varphi \quad \textit{iff} \quad \mathcal{T}^*, h \models \varphi, \quad \textit{for all } h \in H^* \textit{ and } \varphi \in \mathcal{L}_{cc}$$

*Proof of Proposition 4.23.* Consider an arbitrary agent $a_i \in \mathsf{Ag}$ and an arbitrary secret $A_j$ from the set $\mathsf{P}$ of the unique secrets of all the agents. Let $h$ be an arbitrary element of $H^*$. We proceed by induction on $\varphi \in \mathcal{L}_{cc}$.

**Base Case.** Suppose the set of secrets known by $a_i \in \mathsf{Ag}$ at $(G', M')$ is $\mathsf{Q}$. Then $\sim^{M'}_{a_i} = \equiv^{M'}_{\mathsf{Q}}$ (from Definition 3.7). But then also $S_i = \mathsf{Q}$, where, from Definition 4.6, $F(h) = \langle S_1, \ldots, S_n \rangle$ is the gossip situation due to the execution of $h$ at the initial gossip situation. We consider two base cases: $\varphi$ is $Kw_{a_i}A_j$ and $\varphi$ is $\neg Kw_{a_i}A_j$, as follows.

**Case $\varphi$ is $Kw_{a_i}A_j$.**

$$\begin{array}{lll}
G', M' \models_g Kw_{a_i}A_j & \text{iff} \quad A_j \in \mathsf{Q} & \text{(Definition 3.9, Theorem 3.8)} \\
\mathcal{T}^*, h \models Kw_{a_i}A_j & \text{iff} \quad A_j \in S_i & \text{(Given)}
\end{array}$$

Since $S_i = \mathsf{Q}$, we conclude that: $G', M' \models_g Kw_{a_i}A_j \quad \text{iff} \quad \mathcal{T}^*, h \models Kw_{a_i}A_j$

[‡]Here we define the empty history $e$ to be equivalent to the $\mathsf{skip}$ action at the initial gossip situation.

**Case $\varphi$ is $\neg Kw_{a_i}A_j$.**

$$\begin{aligned} G', M' \models_g \neg Kw_{a_i}A_j \quad &\text{iff} \quad G', M' \not\models_g Kw_{a_i}A_j && \text{(Definition 3.9)} \\ &\text{iff} \quad A_j \notin \mathsf{Q} && \text{(Definition 3.9, Theorem 3.8)} \\ &\text{iff} \quad A_j \notin S_i && (S_i = \mathsf{Q}) \\ &\text{iff} \quad \mathcal{T}^*, h \not\models Kw_{a_i}A_j && \text{(Given)} \end{aligned}$$

We therefore conclude that: $G', M' \models_g \neg Kw_{a_i}A_j \quad \text{iff} \quad \mathcal{T}^*, h \models \neg Kw_{a_i}A_j$

**Inductive Hypothesis.** For every $h \in H^*$, if $(G, M_1)[\![h]\!](G', M')$, then for every $\varphi', \varphi'' \in \mathcal{L}_{cc}$, it is the case that:

1. $G', M' \models_g \varphi' \quad$ if and only if $\quad \mathcal{T}^*, h \models \varphi'$
2. $G', M' \models_g \neg\varphi' \quad$ if and only if $\quad \mathcal{T}^*, h \models \neg\varphi'$
3. $G', M' \models_g \varphi'' \quad$ if and only if $\quad \mathcal{T}^*, h \models \varphi''$
4. $G', M' \models_g \neg\varphi'' \quad$ if and only if $\quad \mathcal{T}^*, h \models \neg\varphi''$

**Inductive Step on $\mathcal{L}_{cc}$.** We distinguish the following cases:

**Case $\varphi$ is $\neg\varphi'$.** This case is straightforward from the inductive hypothesis.

**Case $\varphi$ is $(\varphi' \wedge \varphi'')$.** Straightforward from the inductive hypothesis:

$$G', M' \models_g \varphi' \iff \mathcal{T}^*, h \models \varphi' \quad \text{and} \quad G', M' \models_g \varphi'' \iff \mathcal{T}^*, h \models \varphi''$$

And that is equivalent to saying:

$$G', M' \models_g (\varphi' \wedge \varphi'') \iff \mathcal{T}^*, h \models (\varphi' \wedge \varphi'')$$

(semantics of '$\wedge$', propositional logic).

**Case $\varphi$ is $K_{a_i}\varphi'$.**

$$G', M' \models_g K_{a_i}\varphi' \iff \text{for every } N' \text{ such that } M' \approx_{a_i} N', \ G', N' \models_g \varphi' \qquad (\dagger)$$

(from Definition 3.9)

But also,

$$\mathcal{T}^*, h \models K_{a_i}\varphi' \iff \text{for every } h' \text{ such that } h \equiv^*_{a_i} h', \ \mathcal{T}^*, h' \models \varphi' \qquad (\dagger\dagger)$$

(Given)

From the inductive hypothesis we are given that for all $h \in H^*$, if $(G, M_1)[\![h]\!](G', M')$ then $G', M' \models_g \varphi' \iff \mathcal{T}^*, h \models \varphi'$. Consider each of all $h'$ such that $h' \equiv^*_{a_i} h$. We see, from the inductive hypothesis, that if $(G, M_1)[\![h']\!](G', N')$ then $G', N' \models_g$

$\varphi' \iff \mathcal{T}^*, h' \models \varphi'$. Now, suppose $(G, M_1)[\![h']\!](G', N')$, then we conclude that $G', N' \models_g \varphi' \iff \mathcal{T}^*, h' \models \varphi'$, for every $h'$ such that $h' \equiv^*_{a_i} h$. That is:

$$G', N' \models_g \varphi' \text{ for every } h' \text{ such that } h \equiv^*_{a_i} h' \text{ and } (G, M_1)[\![h']\!](G', N') \iff \mathcal{T}^*, h' \models \varphi' \text{ for every } h' \text{ such that } h \equiv^*_{a_i} h' \quad (\dagger\dagger\dagger)$$

(Note that from the definition of $Z^*_{a_m}$ that $h' \equiv^*_{a_i} h$ because $a_i$ knows the same set of secrets in the gossip situations due, respectively to $h$ and $h'$, and this implies (see Corollary 3.11) that $(G', M') \approx_{a_i} (G', N')$, for every $(G', N')$ such that $h' \equiv^*_{a_i} h$ and $(G, M_1)[\![h']\!](G', N')$.)

Therefore, from (†), (††) and (†††), we conclude that:

$$G', M' \models_g K_{a_i}\varphi' \iff \mathcal{T}^*, h \models K_{a_i}\varphi'$$

For the case where $\varphi$ is $(\varphi' \vee \varphi'')$, recall that $(\varphi' \vee \varphi'')$ is equivalent to $\neg(\neg\varphi' \wedge \neg\varphi'')$. Likewise for the case where $\varphi$ is $(\varphi' \to \varphi'')$, recall that $(\varphi' \to \varphi'')$ is equivalent to $\neg(\varphi' \wedge \neg\varphi'')$.

This concludes the inductive argument. □

Notice that the notion of equivalence in Proposition 4.22 is the same as that given in Definition 3.7 and Definition 3.18. In this notion, two gossip situations are equivalent for an agent $a_m$ if and only if $a_m$ knows the same set of secrets in both situations. Notice also that because $H$ is the set of all execution sequences, then in using $\mathcal{T}^*$ to interpret a sequence of calls in a gossip scenario, we allow that the epistemic protocol is *not* common knowledge among the agents, and that each agent can follow a different protocol. (Note that the assumptions of the protocol not being common knowledge and each agent being allowed to follow a different protocol are implicit assumptions behind the interpretation of calls and *complex programs* given in Definition 3.19. We shed these assumptions in the definition of *gossip tree* given in Definition 4.16).

Under the assumptions made in Proposition 4.23, we have shown that if $(G', M')$ is a gossip model due to a call sequence or history $h$, then a property $\varphi \in \mathcal{L}_{cc}$ holds at $(G', M')$ if and only if such property holds at $h$ in the epistemic tree $\mathcal{T}^*$.

**Proposition 4.24.** *Let $\mathcal{T}_g = \langle H, R, F, \{Z_{a_m}\}\rangle$ be the gossip tree for some epistemic gossip protocol $\Pi$. Let $G = (\mathcal{S}, \approx)$ be a gossip model (see Definition 3.7 in Chapter 3), where $M_1$ is the initial gossip situation, and $\mathcal{S} = \{M_1\}$. Let $(G, M_1)[\![h]\!](G', M')$, where $[\![h]\!]$ is as defined in Definition 3.18 and Definition 3.19. Then[§]:*

$$G', M' \models_g \varphi \quad \Longrightarrow \quad \mathcal{T}_g, h \models \varphi, \qquad \textit{for all } h \in H \textit{ and } \varphi \in \mathcal{L}_{cc}$$

[§]Here we define the the empty history $e$ to be equivalent to the skip action at the initial gossip situation.

*Proof of Proposition 4.24.* Consider an arbitrary agent $a_i \in \mathsf{Ag}$ and an arbitrary secret $A_j$ from the set $\mathsf{P}$ of the unique secrets of all the agents. Let $h$ be an arbitrary element of $H$. We proceed by induction on $\varphi \in \mathcal{L}_{cc}$.

**Base Case.** Suppose the set of secrets known by $a_i \in \mathsf{Ag}$ at $(G', M')$ is $\mathsf{Q}$. Then $\sim_{a_i}^{M'} = \equiv_{\mathsf{Q}}^{M'}$ (from Definition 3.7). But then also $S_i = \mathsf{Q}$, where, from Definition 4.6, $F(h) = \langle S_1, \ldots, S_n \rangle$ is the gossip situation due to the execution of $h$ at the initial gossip situation. We consider two base cases: $\varphi$ is $Kw_{a_i}A_j$ and $\varphi$ is $\neg Kw_{a_i}A_j$, as follows.

**Case $\varphi$ is $Kw_{a_i}A_j$.**

$$\begin{array}{lll} G', M' \models_g Kw_{a_i}A_j & \text{iff} \quad A_j \in \mathsf{Q} & \text{(Definition 3.9, Theorem 3.8)} \\ \mathcal{T}_g, h \models Kw_{a_i}A_j & \text{iff} \quad A_j \in S_i & \text{(Definition 4.11)} \end{array}$$

Since $S_i = \mathsf{Q}$, we see that:

$$G', M' \models_g Kw_{a_i}A_j \quad \text{iff} \quad \mathcal{T}_g, h \models Kw_{a_i}A_j$$

And from propositional logic we conclude that:

$$G', M' \models_g Kw_{a_i}A_j \quad \text{implies} \quad \mathcal{T}_g, h \models Kw_{a_i}A_j$$

**Case $\varphi$ is $\neg Kw_{a_i}A_j$.**

$$\begin{array}{lll} G', M' \models_g \neg Kw_{a_i}A_j & \text{iff} \quad G', M' \not\models_g Kw_{a_i}A_j & \text{(Definition 3.9)} \\ & \text{iff} \quad A_j \notin \mathsf{Q} & \text{(Definition 3.9, Theorem 3.8)} \\ & \text{iff} \quad A_j \notin S_i & (S_i = \mathsf{Q}) \\ & \text{iff} \quad \mathcal{T}_g, h \not\models Kw_{a_i}A_j & \text{(Definition 4.11)} \end{array}$$

We therefore see that:

$$G', M' \models_g \neg Kw_{a_i}A_j \quad \text{iff} \quad \mathcal{T}_g, h \models \neg Kw_{a_i}A_j$$

And from propositional logic we conclude that:

$$G', M' \models_g \neg Kw_{a_i}A_j \quad \text{implies} \quad \mathcal{T}_g, h \models \neg Kw_{a_i}A_j$$

**Inductive Hypothesis.** For every $h \in H$, if $(G, M_1)[\![h]\!](G', M')$, then for every $\varphi', \varphi'' \in \mathcal{L}_{cc}$, it is the case that:

1. $G', M' \models_g \varphi' \quad \text{implies} \quad \mathcal{T}_g, h \models \varphi'$
2. $G', M' \models_g \neg\varphi' \quad \text{implies} \quad \mathcal{T}_g, h \models \neg\varphi'$
3. $G', M' \models_g \varphi'' \quad \text{implies} \quad \mathcal{T}_g, h \models \varphi''$
4. $G', M' \models_g \neg\varphi'' \quad \text{implies} \quad \mathcal{T}_g, h \models \neg\varphi''$

**Inductive Step on $\mathcal{L}_{cc}$.** We distinguish the following cases:

**Case $\varphi$ is $\neg\varphi'$.** This case is straightforward from the inductive hypothesis.

**Case $\varphi$ is $(\varphi' \wedge \varphi'')$.** Straightforward from the inductive hypothesis:

$$G', M' \models_g \varphi' \implies \mathcal{T}_g, h \models \varphi' \quad \text{and} \quad G', M' \models_g \varphi'' \implies \mathcal{T}_g, h \models \varphi''$$

And that is equivalent to saying:

$$G', M' \models_g (\varphi' \wedge \varphi'') \implies \mathcal{T}_g, h \models (\varphi' \wedge \varphi'')$$

(semantics of '$\wedge$', propositional logic).

**Case $\varphi$ is $K_{a_i}\varphi'$.**

$$G', M' \models_g K_{a_i}\varphi' \iff \text{for every } N' \text{ such that } M' \approx_{a_i} N', \ G', N' \models_g \varphi' \quad (\dagger)$$

(from Definition 3.9)

But also,

$$\mathcal{T}_g, h \models K_{a_i}\varphi' \iff \text{for every } h' \text{ such that } h \equiv_{a_i} h', \ \mathcal{T}_g, h' \models \varphi' \quad (\dagger\dagger)$$

(from Definition 4.11 and Definition 4.7)

From the inductive hypothesis we are given that for all $h \in H$, if $(G, M_1)[\![h]\!](G', M')$ then $G', M' \models_g \varphi' \implies \mathcal{T}_g, h \models \varphi'$. Consider each of all $h'$ such that $h' \equiv_{a_i} h$. We see, from the inductive hypothesis, that if $(G, M_1)[\![h']\!](G', N')$ then $G', N' \models_g \varphi' \implies \mathcal{T}_g, h' \models \varphi'$. Now, suppose $(G, M_1)[\![h']\!](G', N')$, then we conclude that $G', N' \models_g \varphi' \implies \mathcal{T}_g, h' \models \varphi'$, for every $h'$ such that $h' \equiv_{a_i} h$. That is:

$$G', N' \models_g \varphi' \text{ for every } h' \text{ such that } h \equiv_{a_i} h' \text{ and } (G, M_1)[\![h']\!](G', N') \implies \mathcal{T}_g, h' \models \varphi' \text{ for every } h' \text{ such that } h \equiv_{a_i} h' \quad (\dagger\dagger\dagger)$$

Therefore, from (†), (††) and (†††), we conclude that:

$$G', M' \models_g K_{a_i}\varphi' \implies \mathcal{T}_g, h \models K_{a_i}\varphi'$$

For the case where $\varphi$ is $(\varphi' \vee \varphi'')$, recall that $(\varphi' \vee \varphi'')$ is equivalent to $\neg(\neg\varphi' \wedge \neg\varphi'')$. Likewise for the case where $\varphi$ is $(\varphi' \to \varphi'')$, recall that $(\varphi' \to \varphi'')$ is equivalent to $\neg(\varphi' \wedge \neg\varphi'')$.

This concludes the inductive argument. □

Proposition 4.24 says that if a property is true in a gossip model $(G', M')$ (under the equivalence notion that is based only on what secrets an agent knows in a pair of gossip

situations) then that property is also true at the end of an execution sequence $h$ in the gossip tree $\mathcal{T}_g$ for some protocol $\Pi$, where $h$ is the same execution sequence that gave rise to $(G', M')$ after being executed at the initial gossip situation $(G, M_1)$.

**Proposition 4.25.** *For any epistemic gossip protocols* $\Pi$ *and* $\Pi'$*, let* $\varphi_\Pi(x, y)$ *be the epistemic calling condition for agent* $x$ *to call agent* $y$ *for Protocol* $\Pi$ *and let* $\varphi_{\Pi'}(x, y)$ *be the epistemic calling condition for agent* $x$ *to call agent* $y$ *for Protocol* $\Pi'$*, where* $x, y \in \mathsf{Ag}$*. Suppose* $\mathcal{T}_g^\Pi = \langle H, R, F, \{Z_{a_m}\}\rangle$ *is the gossip tree for protocol* $\Pi$*, and* $\Sigma(\Pi)$ *and* $\Sigma(\Pi')$ *are, respectively, the extension of* $\Pi$ *and* $\Pi'$*, then:*

$$\mathcal{T}_g^\Pi \models \varphi_\Pi(x, y) \to \varphi_{\Pi'}(x, y) \text{ implies } \Sigma(\Pi) \subseteq \Sigma(\Pi')$$

*Proof.* Consider an arbitrary history $h \in H$. Suppose that $\mathcal{T}_g^\Pi, h \models \varphi_\Pi(x, y) \to \varphi_{\Pi'}(x, y)$. Then it follows that if a call $xy$ can be made for protocol $\Pi$, then the same call can also be made for protocol $\Pi'$, after the execution sequence given by $h$ (that is, if $h; xy \in \Sigma(\Pi)$ then $h; xy \in \Sigma(\Pi')$). Since the foregoing statement is true for an arbitrary history $h \in H$, it follows that any execution sequence in $\Sigma(\Pi)$ is also in $\Sigma(\Pi')$. And therefore $\Sigma(\Pi) \subseteq \Sigma(\Pi')$. □

*Observation* 4.26. Recall Proposition 3.27, as follows. Let $\Pi_0$ denote the Fixed Schedule:

1. $\Sigma(\Pi_1) \subsetneq \Sigma(\Pi_3) \subsetneq \Sigma(\Pi_2) \subsetneq \Sigma(\Pi_5) = \Sigma(\Pi_4)$
2. $\Sigma(\Pi_0) \not\subseteq \Sigma(\Pi_1)$ and $\Sigma(\Pi_0) \subsetneq \Sigma(\Pi_3)$ and $\Sigma(\Pi_0) \subsetneq \Sigma(\Pi_2)$ and $\Sigma(\Pi_0) \subsetneq \Sigma(\Pi_4)$ and $\Sigma(\Pi_0) \subsetneq \Sigma(\Pi_5)$

The proof of Proposition 3.27 (see Page 72) was done by using Proposition 3.26, and then arguing for the following seven claims, where $cc(x, y, \Pi_i) = \varphi_{\Pi_i}(x, y)$:

Claim 1: $\models_g cc(x, y, \Pi_1) \to cc(x, y, \Pi_3)$

Claim 2: $\not\models_g cc(x, y, \Pi_3) \to cc(x, y, \Pi_1)$

Claim 3: $\models_g cc(x, y, \Pi_3) \to cc(x, y, \Pi_2)$

Claim 4: $\not\models_g cc(x, y, \Pi_2) \to cc(x, y, \Pi_3)$

Claim 5: $\models_g cc(x, y, \Pi_2) \to cc(x, y, \Pi_5)$

Claim 6: $\not\models_g cc(x, y, \Pi_5) \to cc(x, y, \Pi_2)$

Claim 7: $\models_g cc(x, y, \Pi_5) \leftrightarrow cc(x, y, \Pi_4)$

Let $\mathcal{T}_g^\Pi = \langle H, R, F, \{Z_{a_m}\}\rangle$ and let $h$ be an arbitrary element of $H$. Observe that Proposition 4.25 can also be used to prove Proposition 3.27 by arguing for the following claims:

Claim 1': $\mathcal{T}_g^\Pi \models cc(x, y, \Pi_1) \to cc(x, y, \Pi_3)$

Claim 2': $\mathcal{T}_g^{\Pi} \not\models cc(x, y, \Pi_3) \rightarrow cc(x, y, \Pi_1)$

Claim 3': $\mathcal{T}_g^{\Pi} \models cc(x, y, \Pi_3) \rightarrow cc(x, y, \Pi_2)$

Claim 4': $\mathcal{T}_g^{\Pi} \not\models cc(x, y, \Pi_2) \rightarrow cc(x, y, \Pi_3)$

Claim 5': $\mathcal{T}_g^{\Pi} \models cc(x, y, \Pi_2) \rightarrow cc(x, y, \Pi_5)$

Claim 6': $\mathcal{T}_g^{\Pi} \not\models cc(x, y, \Pi_5) \rightarrow cc(x, y, \Pi_2)$

Claim 7': $\mathcal{T}_g^{\Pi} \models cc(x, y, \Pi_5) \leftrightarrow cc(x, y, \Pi_4)$

Based on the argument for Claims 1 through 7 (see proof of Proposition 3.27 on Page 72), we obtain the argument for Claim 1' through Claim 7', respectively, by substituting $(\mathcal{T}_g^{\Pi}, h)$ for the arbitrary gossip model $(G, M)$.

**Automatic Construction of a Gossip Tree**

In order to construct layer $\lambda_{k+1}$ of a gossip tree, only layer $\lambda_k$ is required. So, given the initial layer $\lambda_0$ of the gossip tree of a specified protocol, it possible to automatically generate the entire gossip tree, layer by layer. Algorithm 4.1 generates the gossip tree for a given epistemic gossip protocol $\Pi$. Given a layer $\lambda_k$ of a gossip tree, the *ComputeNext-Layer* function constructs the next layer $\lambda_{k+1}$ of the gossip tree, whereas the *LayerLabel* function updates the layer label $L_{a_i}(\lambda_{k+1})$ of the gossip tree layer under construction, for all $a_i \in \mathsf{Ag}$. Beginning with an initial layer, Algorithm 4.1 builds the gossip tree up to a desired finite layer. Given a layer of the gossip tree, the algorithm computes all the calls that are possible at each node of the given layer, by model checking the epistemic calling condition $\varphi_{\Pi}(a_i, a_j) = \varphi_{a_i a_j}$ for each pair of agents $(a_i, a_j)$, at the node. For each possible call at a node, the algorithm produces a successor node which is naturally in the next layer from the layer of the given node. Algorithm 4.1 exploits the following properties of gossip trees:

**Proposition 4.27.** *Given any two layers $\lambda_k$ and $\lambda_{k+1}$ of a gossip tree $\mathcal{T}_g$, and given any $h' \in \lambda_k$ and $h \in \lambda_{k+1}$, then $h'Rh$ iff $Z_{a_i}(h')R_cZ_{a_i}(h)$, for all $a_i \in \mathsf{Ag}$.*

*Proof.* Let $p(h) = \hat{h}$, such that $\hat{h}Rh$. Choose an arbitrary agent $a_i \in \mathsf{Ag}$.

$\Longrightarrow$ **-direction.** Suppose $h'Rh$. Consider an arbitrary $\bar{h} \in Z_{a_i}(h)$. One of the conditions for equivalence of $\bar{h}$ and $h$ is that $p(h) \equiv_{a_i} p(\bar{h})$ (from Definition 4.7). But $h \in Z_{a_i}(h)$ and $p(h) = h'$, so $h' \equiv_{a_i} p(\bar{h})$. Since $\bar{h}$ is an arbitrary element of $Z_{a_i}(h)$, and from the definition of $R_c$ in Definition 4.9, we conclude that $Z_{a_i}(h')R_cZ_{a_i}(h)$.

$\Longleftarrow$ **-direction.** Suppose $Z_{a_i}(h')R_cZ_{a_i}(h)$. Choose an arbitrary $\bar{h} \in Z_{a_i}(h)$, then there must be a $\bar{h}' \in Z_{a_i}(h')$ such that $\bar{h}'R\bar{h}$ (from the definition of $R_c$ in Definition 4.9). For every $h'' \in Z_{a_i}(h')$, $Z_{a_i}(h'') = Z_{a_i}(h')$. We chose $\bar{h}$ arbitrarily, so now we fix a node $h \in Z_{a_i}(h)$ and call its parent $h'$. Then $Z_{a_i}(h')R_cZ_{a_i}(h)$ implies $h'Rh$. □

**Algorithm 4.1** Automatic construction of a gossip tree layer.

**function** ComputeNextLayer($\lambda_k$, $\varphi_{ab}$)
  $\mathcal{C} \leftarrow \{a_i a_j \mid a_i, a_j \in \mathsf{Ag}, a_i \neq a_j\}$
  **if** $\lambda_k = \emptyset$ **then** $\lambda_k \leftarrow \{e\}$, $L_{a_i}(\lambda_k) \leftarrow \emptyset$ and $L_{a_i}(\lambda_k) \leftarrow L_{a_i}(\lambda_k) \cup \{\{e\}\}$, $\forall a_i \in \mathsf{Ag}$ **end if**
  $L_{a_i}(\lambda_{k+1}) \leftarrow \emptyset$
  **for all** $h' \in \lambda_k$ **do**
    **for all** $a_i a_j \in \mathcal{C}$ **do**
      **if** SAT($\varphi_{a_i a_j}, h'$) *holds* **then**
        $h \leftarrow h'; ab$, and $F(h)$ is computed accordingly
        LayerLabel($\lambda_{k+1}, h', h$)
      **end if**
    **end for**
  **end for**
  **return** $\lambda_{k+1}$
**end function**

**function** LayerLabel($\lambda_{k+1}, h', h$)
  **for all** $\mathbf{a_i} \in \mathsf{Ag}$ :
    if $\exists C' = Z_{a_i}(h'')$ such that $h' R\, h$ and $Z_{a_i}(h') R_c\, Z_{a_i}(h'')$ and $h \equiv_{a_i} h''$
      $C' \leftarrow C' \cup \{h\}$
    else
      Initialise an empty cell $C$
      $C \leftarrow C \cup \{h\}$
      $L_{a_i}(\lambda_{k+1}) \leftarrow L_{a_i}(\lambda_{k+1}) \cup \{C\}$
    end if
  **end for**
**end function**

**Proposition 4.28.** *Let $h$ and $h'$ be as in Proposition 4.27. If $h$ is assigned to any child cell of $Z_{a_i}(h')$, then $h$ is not in any other child cell of $Z_{a_i}(h')$.*

*Proof.* Towards a contradiction, choose an arbitrary agent $a_i$ from the given set of agents. Let $C'$ and $C''$ be two distinct child cells of $Z_{a_i}(h')$. Suppose that $h$ is assigned to $C'$ and $h$ is also assigned to $C''$, then it follows that for all $\bar{h} \in C'$ and for all $\bar{\bar{h}} \in C''$, $h \equiv_{a_i} \bar{h} \equiv_{a_i} \bar{\bar{h}}$. Then from the definition of cell it follows that $C' = C''$ for agent $a_i$ and therefore $C'$ and $C''$ are not distinct, contrary to the assumption. □

Given a parent node $h'$ and its successor $h$, the LayerLabel function assigns $h$ to a cell within the partition for each agent, as follows. Consider the partition for an arbitrary agent $a_i \in \mathsf{Ag}$. Since the cell of $h$, for agent $a_i$, is a child cell of $Z_{a_i}(h')$ (Proposition 4.27), the LayerLabel function first checks whether $Z_{a_i}(h')$ already has any child cells. Let such a child cell be called $C'$. The node $h$ can be assigned to such $C'$ if there exists an $h'' \in C'$ such that $h'' \equiv_{a_i} h$. If $h$ cannot be assigned to any currently existing child cell of $Z_{a_i}(h')$ then we create a new empty child cell $C$ for $Z_{a_i}(h')$ and assign $h$ to it. Moreover, LayerLabel function ensures that $h$ is assigned to only one cell (Proposition 4.28).

Furthermore, for any $h \in \lambda_{k+1}$, where $k \geq 0$, the condition under which $h$ is assigned to a cell is equivalent to that given in Definition 4.7. Given $\lambda_{k+1}$, and given $h'$ and $h$ such that $h'Rh$, where $h \in \lambda_{k+1}$, the history $h$ is assigned to some $C' = Z_{a_i}(h'')$ such that $Z_{a_i}(h')R_cZ_{a_i}(h'')$ and $h \equiv_{a_i} h''$. Again let $p(h) = \hat{h}$, such that $\hat{h}Rh$. The condition $\eta = (h'Rh$ and $Z_{a_i}(h')R_cZ_{a_i}(h''))$ implies that $p(h'') \equiv_{a_i} h' = p(h)$, since $p(h'') \in Z_{a_i}(h')$ by the definition of $R_c$. Therefore $\eta$ ensures that $p(h) \equiv_{a_i} p(h'')$, which is required by Definition 4.7. Based on the fact that $p(h) \equiv_{a_i} p(h'')$, the condition $h \equiv_{a_i} h''$ is then checked according to Definition 4.7.

### The SAT Function

Our model checking algorithm combines the bottom-up approach and the top-down approach similar to that employed in temporal-epistemic model checking (see Algorithm 4.4). In the bottom-up approach [33], a given formula $\varphi$ is checked in a state of the model by iteratively obtaining all the states where the subformulas of $\varphi$ are true, beginning with the smallest subformula of $\varphi$, and increasing the size of the subformulas in a step-wise manner in each iteration, until the set of states satisfying the largest subformula, namely $\varphi$ itself, is obtained. Each higher subformula of $\varphi$ is checked on the states obtained from the previous iteration. This approach is called the bottom-up approach because model checking starts with the smallest subformula of $\varphi$. In the top-down approach, the reverse is the case. At the given state of the model, $\varphi$ is checked by recursively checking its subformulas in order of decreasing size until the smallest subformula is checked [55]. Our model checking algorithm is with respect to the language $\mathcal{L}_{cc}$. In the general temporal and epistemic setting, the top-down approach is more computationally expensive than the bottom-up approach. However, in our experiments we show that by combining the top-down and the bottom-up approach we take advantage of the peculiarities of the equivalence classes obtained from the gossip tree to obtain a better performance in practice, than by using the bottom-up approach (see subsection *Equivalence Class Analysis*, later). We also obtain added performance by means of our representation of the gossip tree and the layer labelling procedure, which introduces a caching technique for faster computation of equivalence classes by reusing equivalence class information from previous rounds.

We now describe our SAT function as used in Algorithm 4.1, but first we give some ancillary definitions as follows:

**Definition 4.29** (Model Checking)**.** Given $h \in \lambda_k$ of a gossip tree $\mathcal{T}_g$, and a formula $\varphi \in \mathcal{L}_{cc}$, the model checking problem is whether $\varphi$ is satisfied at $h$, i.e., whether $\lambda_k, h \models \varphi$. The output is "yes" if $\varphi$ is satisfied at the given $h$, and otherwise "no". The model checking algorithm is defined in Algorithm 4.4.

**Definition 4.30** (Relevant Set)**.** Let the language $\mathcal{L}'_{cc}$ be equal to $\mathcal{L}_{cc}$ without the fragment $K_{a_i}\varphi$. That is:

$$\mathcal{L}'_{cc} \ni \quad \varphi \quad ::= \quad Kw_{a_i}A_j \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \rightarrow \varphi) \ , \text{ where } a_i \neq a_j.$$

Let $\varphi' \in \mathcal{L}'_{cc}$ be called *epistemic propositional formula*. Given a formula $\varphi \in \mathcal{L}_{cc}$, and $h \in \lambda_k$, let Relevant Set $QQ$ with respect to $\varphi$ and $h$ be defined in Algorithm 4.2, with $QQ = \{h\}$, initially.

Intuitively, the relevant set is the largest set of gossip tree nodes needed for the model checking of $\varphi$ at the designated node $h$.

**Definition 4.31** (Truth Set)**.** The Truth Set $TT$ is defined in Algorithm 4.3, with $TT = \emptyset$, initially.

**Algorithm 4.2** Definition of Relevant Set.

**function** TopDownSAT($\varphi, QQ$)
  **begin case**
    $\varphi$ is epistemic propositional: **return** $QQ$
    $\varphi$ is $Ka_i\varphi_1$:
      $QQ' \leftarrow \emptyset$
      for every $h' \in QQ$
        $QQ' \leftarrow QQ' \cup Z_{a_i}(h')$
      end for
      $QQ \leftarrow QQ'$
      TopDownSAT($\varphi_1, QQ$)
    $\varphi$ is $\neg\varphi_1$:
      $QQ \leftarrow$ TopDownSAT($\varphi_1, QQ$)
    $\varphi$ is $(\varphi_1 \vee \varphi_2)$:
      $QQ \leftarrow$ TopDownSAT($\varphi_1, QQ$) $\cup$ TopDownSAT($\varphi_2, QQ$)
    $\varphi$ is $(\varphi_1 \wedge \varphi_2)$:
      $QQ \leftarrow$ TopDownSAT($\varphi_1, QQ$) $\cup$ TopDownSAT($\varphi_2, QQ$)
    $\varphi$ is $(\varphi_1 \rightarrow \varphi_2)$:
      $QQ \leftarrow$ TopDownSAT($(\neg\varphi_1 \vee \varphi_2), QQ$)
  **end case**
**end function**

Here, the *TopDownSAT* function mitigates the state-space explosion by narrowing down the set of nodes to relevant ones. We then employ the *BottomUpSAT*, which is an adaptation of the CTL labelling algorithm, on the relevant set. We show that this two-step approach is better in practice than the bottom-up or top-down approach.

We sketch an argument to establish the correctness of the SAT function as follows. From the semantics of $\mathcal{L}_{cc}$ it is easy to see that *TopDownSAT* computes the set of all nodes needed to model-check the given formula $\varphi$ on the designated node $h$. By the case basis, if $\varphi$ is epistemic propositional, then we need only the designated node $h$; if $\varphi$ is $K_{a_i}\varphi'$ then we need the set $QQ'$ of nodes that agent $a_i$ cannot distinguish from each of the nodes contained in $QQ$, in order to check whether $K_{a_i}\varphi'$. Note that due to the reflexivity property of the accessibility relation (a node is equivalent to itself, for all the agents) $QQ \subseteq QQ'$. If $\varphi$ is $\neg K_{a_i}\varphi'$ then we need *at most* the same nodes as for $K_{a_i}\varphi$. If $\varphi$ is $(\varphi' \wedge \varphi'')$ or $(\varphi' \vee \varphi'')$, we need the set of relevant nodes for $\varphi'$ *union* the set of relevant nodes for $\varphi''$.

**Algorithm 4.3** Definition of Truth Set.

**function** BottomUpSAT($\varphi, TT, QQ$)
  **begin case**
    $\varphi$ is epistemic propositional:
      $TT \leftarrow \{h |\ h \in QQ$ and $\varphi$ is true at $h\}$
      **return** $TT$
    $\varphi$ is $K_{a_i}\varphi_1$:
      $TT \leftarrow SAT_k(a, \varphi_1, TT, QQ)$
      **return** $TT$
    $\varphi$ is $\neg\varphi_1$:
      $TT \leftarrow QQ \setminus$ BottomUpSAT($\varphi_1, TT, QQ$)
      **return** $TT$
    $\varphi$ is $(\varphi_1 \vee \varphi_2)$:
      $TT \leftarrow$ BottomUpSAT($\varphi_1, TT, QQ$) $\cup$ BottomUpSAT($\varphi_2, TT, QQ$)
      **return** $TT$
    $\varphi$ is $(\varphi_1 \wedge \varphi_2)$:
      $TT \leftarrow$ BottomUpSAT($\varphi_1, TT, QQ$) $\cap$ BottomUpSAT($\varphi_2, TT, QQ$)
      **return** $TT$
    $\varphi$ is $(\varphi_1 \rightarrow \varphi_2)$:
      $TT \leftarrow$ BottomUpSAT($(\neg\varphi_1 \vee \varphi_2), TT, QQ$)
      **return** $TT$
  **end case**
**end function**

**function** $SAT_k(a, \varphi, TT, QQ)$
  $TT' \leftarrow \emptyset$
  for every $h \in QQ$:
    if $Z_{a_i}(h) \subseteq TT$, then $TT' \leftarrow TT' \cup \{h\}$
  end for
  **return** $TT'$
**end function**

To prove the correctness of the *BottomUpSAT* function, we note that in order to check the formula $\varphi$ on the given node $h$ we do not require any node that is not in $QQ$, as returned by *TopDownSAT*.

### Time Complexity of the SAT Function

We come now to discuss the time complexity of the SAT function. But first we give some definitions.

**Definition 4.32** (Epistemic Depth)**.** We define the epistemic depth of a formula $\varphi \in \mathcal{L}_{cc}$ as follows. Let $\mathcal{D}(\varphi)$ be a function that computes the epistemic depth of $\varphi$, then:

$$\begin{aligned}
\mathcal{D}(Kw_{a_i}A_j) &= 0 \\
\mathcal{D}(\neg\varphi) &= \mathcal{D}(\varphi) \\
\mathcal{D}((\varphi_1 \wedge \varphi_2)) &= max(\mathcal{D}(\varphi_1), \mathcal{D}(\varphi_2)) \\
\mathcal{D}((\varphi_1 \vee \varphi_2)) &= max(\mathcal{D}(\varphi_1), \mathcal{D}(\varphi_2)) \\
\mathcal{D}((\varphi_1 \rightarrow \varphi_2)) &= max(\mathcal{D}(\varphi_1), \mathcal{D}(\varphi_2)) \\
\mathcal{D}(K_{a_i}\varphi) &= 1 + \mathcal{D}(\varphi)
\end{aligned}$$

**Definition 4.33** (Iterated-K Subformula)**.** An iterated-K subformula is a subformula of the language $\mathcal{L}_{itk}$ such that $\varphi ::= Kw_{a_i}A_j \,|\, (\varphi \wedge \varphi) \,|\, (\varphi \vee \varphi) \,|\, \neg\varphi$, and $\mathcal{L}_{itk} \ni \psi ::= K_{a_i}\varphi \,|\, \neg\psi$. The length of an iterated-$K$ subformula $\psi$ is the epistemic depth of $\psi$.

Intuitively, the epistemic depth of the subformula $\psi$ is the number of $K$ operators (that is, $K$ or $\neg K$ operator) in $\psi$.

**Proposition 4.34.** *The time complexity of the SAT function shown in Algorithm 4.4 is $O((m + |Ag|).|\lambda_k|^2)$, where $m$ is the epistemic depth of the given formula $\varphi \in \mathcal{L}_{cc}$.*

Considering *TopDownSAT* (shown in Algorithm 4.2), we obtain the equivalence class of a node for an agent in constant time, and for each $K$ operator in an iterated-K formula, the relevant set is computed in time $O(|\lambda_k|)$: for the "leftmost" $K$ operator, at the designated node $h$, we can obtain $QQ = Z_{a_i}(h)$ in constant time, where $a_i$ is the agent associated with the $K$ operator; for each subsequent $K$ operator, left-to-right, we compute $Z_{a_i}(h')$ for each $h' \in QQ$ in time $O(|QQ|)$ which is in turn at most $|\lambda_k|$. For $m$ iterated-$K$ operators we compute $QQ$ in $O(1) + O((m-1).|\lambda_k|) = O((m-1).|\lambda_k|)$. The time complexity of model-checking an epistemic propositional formulae is $O(|\lambda_k|)$ since we would have to check all the nodes in the relevant set, which are at most $|\lambda_k|$ nodes. For a conjunctive or disjunctive formula the worst-case complexity is that of its most expensive term. Therefore the time complexity of a given $\varphi \in \mathcal{L}_{cc}$ is the complexity of the longest iterated-K subformula, that is, the iterated-K subformula with the most number of $K$ operators.

Considering *BottomUpSAT* (shown in Algorithm 4.3), we compute the truthset $TT$ in time $O(|\lambda_k|)$, and for each non-negated iterated-$K$ operator ($K_{a_i}\varphi'$) associated with some agent $a_i \in \mathsf{Ag}$, we compute the truthset $TT'$ of all nodes $h$ in $QQ$ whose equivalence class is a subset of $TT$, in time $O(|\lambda_k|)$ for each node $h$; this gives $O(|\lambda_k|^2)$ for all nodes in $QQ$. For each negated iterated-$K$ operator ($\neg K_{a_i}\varphi$) we compute $TT'$ as for the non-negated iterated-$K$ operator, and take the complement of the obtained $TT'$. So, the time complexity of *BottomUpSAT* algorithm is $O(m.|\lambda_k|^2)$. Considering the *LayerLabel* algorithm, we create $|\lambda_k|$ nodes, and update the layer label, for each of the nodes created and for each of the agents. We process each node $h'$ as follows: the parent cell is retrieved in constant time; child cells of the parent cell are also retrieved in constant time; we compare a member from each child cell, with $h'$, to determine membership of

$h'$. This is achieved in $(|\lambda_k|)$. Inclusion of $h'$ into a cell is done in constant time. This gives a time complexity of $O(|Ag|.|\lambda_k|^2)$ for the *LayerLabel* algorithm.

We will see in the next section that, given the nature of gossip models, the size of the equivalence classes are exponentially small compared to the size of the $\lambda_k$, and as such, the actual size of the relevant set is small compared to the size of $\lambda_k$, thus lending space and time efficiency in practical terms.

**Algorithm 4.4** The SAT function.

```
1: function SAT(φ, h)
2:     QQ' ← h
3:     TT' ← ∅
4:     QQ ← TopDownSAT(φ, QQ')
5:     TT ← BottomUpSAT(φ, TT', QQ)
6:     if h ∈ TT then return true          ▷ true is "Yes"
7:     else return false                   ▷ false is "No"
8:     end if
9: end function
```

### Equivalence Class Analysis

In this section we investigate the sizes of the equivalence classes for the agents at each layer of the gossip tree model. We show that the *TopDownSAT* could be much less expensive in practice, due to the relatively small sizes of equivalence classes for the agents in each layer.

The data presented in Tables 4.1 and 4.2 is the summary of an equivalence class analysis for one of five agents in Protocol 2 and 3 respectively, beginning from the root layer to the terminal layer of the gossip tree. We chose only one of the agents because we observed that the cell sizes (and their distribution) for other agents are symmetrical variants of each other (see examples in Figures 4.3 and 4.4). From Tables 4.1 and 4.2 we observe that the cell sizes in each layer are indeed very small compared to the layer size. The graph in Figure 4.5 shows that the average cell size is exponentially smaller than the layer size for both Protocol 2 and Protocol 3. The same trend is found when the experiment is repeated for three and four agents. We did not carry out the analysis for Protocol 1 because, strictly speaking, to check its epistemic calling condition on a given situation we require only the information contained in that same situation, namely the secrets known by the agents in the situation - hence there is no need to reason about other possible worlds.

The results shown in Tables 4.1 and 4.2 also indicate that indeed the relevant set obtained through the *TopDownSAT* is significantly small compared to the size of a layer, and hence lends added performance to the *SAT* function in practice. To illustrate this point consider our example protocols (protocols 1-5). The epistemic depth of the epistemic calling condition for each of these protocols is at most one. Particularly, consider the equivalence class analysis for Protocol 2 for a scenario with five agents

(see Table 4.1). Consider layer 10 of the gossip tree of this protocol and assume that the size of the equivalence class required to check the satisfiability of the epistemic calling condition is the average cell size of that layer, that is 1920 nodes. Then instead of checking $472,988,160$ nodes to determine the nodes that satisfy the propositional part of the epistemic calling condition (as required by the solely bottom-up approach), we would rather check only 1920 nodes, yielding a 99.9996% reduction in the number of nodes checked. Furthermore, consider an epistemic gossip protocol whose epistemic calling condition has an epistemic depth of two. Let us reuse the figures as in the previous example and assume that the equivalence classes are disjoint. Then the relevant set for such epistemic calling condition comprises of $1920^2 = 3,686,400$ nodes, which still yields a 99.2206% reduction in the number of nodes checked.

Finally, as expected (see Proposition 3.27 and Observation 4.26), the layer sizes for Protocol 2 are mostly larger than those of the corresponding layers of Protocol 3. In Figure 4.6 we see that between layer 0 and 5, the two protocols exhibit similar layer sizes. However layer 5 through 8 shows a linear growth in the size of layers of Protocol 2 relative to those of Protocol 3. The relative size of the layers then remained fairly constant from layer 8 through 10 for both protocols.

$$
\begin{aligned}
L_a(\lambda_1): &\quad 2,2,2,6\\
L_b(\lambda_1): &\quad 2,6,2,2\\
L_c(\lambda_1): &\quad 6,2,2,2\\
L_d(\lambda_1): &\quad 6,2,2,2
\end{aligned}
$$

FIGURE 4.3: **The first layer of the gossip tree of Protocol 2 (four-agent scenario).**

$$
\begin{aligned}
L_a(\lambda_2): &\quad 4,4,4,4,4,4,4,4,4,4,12,12,4,24,4,12,4,4,4\\
L_b(\lambda_2): &\quad 12,4,4,4,24,12,4,12,4,4,4,4,4,4,4,4,4,4,4\\
L_c(\lambda_2): &\quad 4,12,24,4,4,12,4,4,4,4,4,4,4,4,4,12,4,4,4\\
L_d(\lambda_2): &\quad 24,12,12,4,4,4,4,4,4,4,4,4,4,4,12,4,4,4,4
\end{aligned}
$$

FIGURE 4.4: **The second layer of the gossip tree of Protocol 2 (four-agent scenario).**

TABLE 4.1: **Equivalence class summary for Protocol 2 (five-agent scenario).**

| k | $\lvert\lambda_k\rvert$ | Number of Cells | Min. Cell Size | Max. Cell Size | Average Cell Size |
|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 20 | 5 | 2 | 12 | 4 |
| 2 | 360 | 33 | 4 | 120 | 11 |
| 3 | 6,000 | 217 | 4 | 1,056 | 28 |
| 4 | 86,880 | 1,161 | 8 | 6,912 | 75 |
| 5 | 1,016,640 | 5,221 | 8 | 25,920 | 195 |
| 6 | 9,063,360 | 19,653 | 8 | 52,224 | 461 |
| 7 | 47,178,240 | 59,968 | 16 | 106,368 | 787 |
| 8 | 155,675,520 | 151,192 | 16 | 269,472 | 1,030 |
| 9 | 316,947,840 | 261,976 | 16 | 588,384 | 1,210 |
| 10 | 472,988,160 | 246,388 | 16 | 714,240 | 1,920 |

TABLE 4.2: **Equivalence class summary for Protocol 3 (five-agent scenario).**

| k | $\lvert\lambda_k\rvert$ | Number of Cells | Min. Cell Size | Max. Cell Size | Average Cell Size |
|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 20 | 5 | 2 | 12 | 4 |
| 2 | 360 | 33 | 4 | 120 | 11 |
| 3 | 6,000 | 217 | 4 | 1,056 | 28 |
| 4 | 86,880 | 1,161 | 8 | 6,912 | 75 |
| 5 | 993,600 | 5,029 | 8 | 23,232 | 198 |
| 6 | 7,764,480 | 17,325 | 8 | 44,448 | 448 |
| 7 | 36,969,600 | 48,556 | 16 | 80,256 | 761 |
| 8 | 107,021,392 | 108,655 | 16 | 160,512 | 985 |
| 9 | 239,439,360 | 190,312 | 16 | 321,024 | 1,258 |
| 10 | 325,891,200 | 167,644 | 16 | 617,472 | 1,944 |

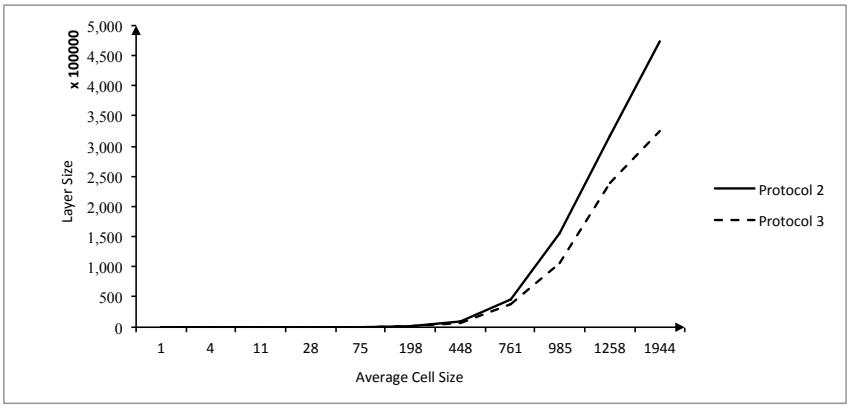



FIGURE 4.5: **Average cell size versus layer size.**

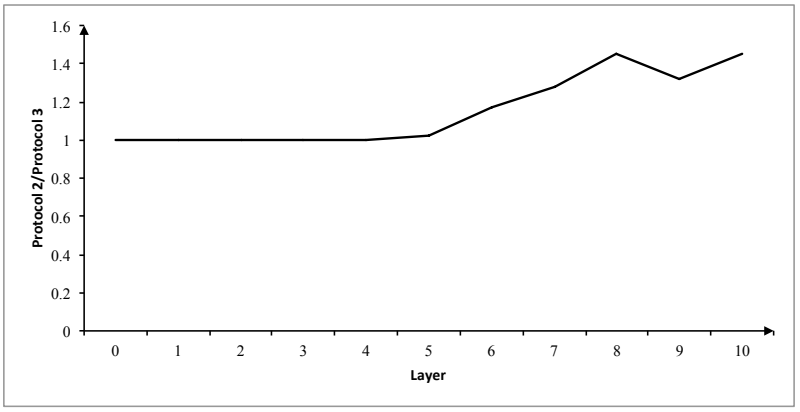



FIGURE 4.6: **Comparing layer sizes of Protocol 2 to those of Protocol 3.**

## 4.3 Implementation Notes

We now present some implementation specific notes. We discuss how we capture the details of the extension of an epistemic gossip protocol, and we present an approach we employed for parallel processing during execution of the EGP tool.

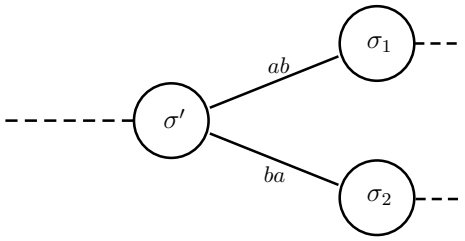
### 4.3.1 Protocol Extension

We adopt the standard extension as the basis for the comparison of the extension size of terminating epistemic gossip protocols. We measure the size of the standard extension of an epistemic gossip protocol by counting the number of leaf nodes in the gossip tree for the protocol, since this corresponds to the number of execution sequences that start from the initial gossip situation and reach termination. An execution sequence terminates either because it has achieved the goal state that every agent knows every other agent's secret, or due to a deadlock in the resulting gossip situation due to the calling condition not being satisfied for any pair of agents to call each other although the goal state have not been reached. We also allow that an execution sequence can terminate due to a stoppage by the user of the EGP tool. When an execution sequence terminates, we record its length and note whether it is successful or not. Finally we count all the execution sequences of each length value, and calculate the average execution length of the protocol.

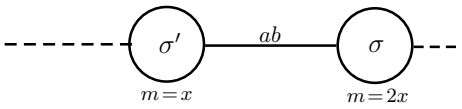
In order to capture all the execution sequences in a protocol's extension, we need to explicitly keep all the execution sequences in memory during protocol execution. But this can be very demanding with respect to the computing time and memory resources especially with increasing number of agents in the scenario. However, we do not need to keep all the execution sequences in memory in order to: measure the size of the protocol's standard extension, take the count of terminal execution sequences of various length values or calculate the average execution length of the protocol from the terminal execution sequences. Therefore in order to allow for various needs of the user of the EGP tool, we enable the user to run the tool in two modes, namely: (a) sparse tree mode, and (b) full tree mode. The sparse tree mode allows the protocols to run faster and with less memory, and gives results for the average execution length, the count of various terminal execution sequence length values and the standard extension size of the protocol, although the user will not obtain all the execution sequences in the protocols standard extension using this mode. The full standard extension of the protocol is obtained by using the full tree mode which is less memory and time efficient than the sparse tree mode.

Let us now briefly discuss the technique we use for the sparse tree mode.

Consider a branch of a gossip tree shown in Figure 4.7(a). The node $\sigma'$ is the parent node of $\sigma_1$ and $\sigma_2$. The node $\sigma_1$ is due to a call $ab$ at $\sigma'$, whereas the node $\sigma_2$ is due to a call $ba$ at $\sigma'$. It is obvious that all the agents know the same secrets at the gossip situation resulting from $\sigma_1$ as they do for the gossip situation resulting from $\sigma_2$. When




(a) A Branch of a Full Tree




(b) A Branch of a Sparse Tree

FIGURE 4.7: **Jointed versus non-jointed nodes.**

such pair of calls $ab$ and $ba$ occur at the same node of a gossip tree we call them *mirrored calls*. Recall that we distinguish between the call $ab$ and the call $ba$. One reason for this distinction is that under the epistemic gossip protocols both calls are not always jointly possible. For example, the epistemic calling condition may be satisfied for agent $a$ to call agent $b$, but the epistemic calling condition for agent $b$ to call agent $a$ is not satisfied. More concretely, take the Learn New Secrets protocol where an agent $a$ can call another agent $b$ if agent $a$ does not know the unique secret of agent $b$. Now consider the following execution sequence: $bc; ac; \ldots$, after the first two calls agent $a$ can no longer call agent $b$ whose unique secret it learnt from agent $c$, but then agent $b$ can still call agent $a$ because $b$ have not yet learnt the unique secret of agent $a$. That said, it is clear that the extension of a protocol will not be fully captured if the sequence $\sigma_1$ were always identified with the sequence $\sigma_2$. However, in the sparse tree mode, for the purpose of increasing time and memory efficiency, we identify two such nodes $\sigma_1$ and $\sigma_2$, that is, instead of creating two child nodes as a result of the mirrored calls at $\sigma'$, we create only one child node $\sigma$ as a result of the mirrored calls. Let us then say we have *joined* such $\sigma_1$ and $\sigma_2$ into $\sigma$, and refer to such $\sigma$ as a *jointed* node, and also refer to such $\sigma_1$ and $\sigma_2$ as the *jointees* of $\sigma$. Then, we keep track of the *multiplicity value* of the nodes as follows. The root node of a gossip tree has a multiplicity value of one. When a child node is produced from a parent node, the child node's multiplicity initially takes the value of the multiplicity of its parent node. If a child node is a jointed node, then its multiplicity value becomes the value of its parent's multiplicity multiplied by two. See Figure 4.7(b) for the jointed version of the branch shown in Figure 4.7(a). (Note that in Figure 4.7(b), $m$ stands for the value of the multiplicity of a node).

Therefore in the sparse tree mode, when an execution sequence terminates, we increment the size of the standard extension by the value of the multiplicity of the terminal (or leaf) node of that execution sequence. We now prove that the sparse tree representation does not alter the satisfiability of the epistemic calling conditions when compared

with the full tree representation. Moreover, we show that the same standard extension size and counts of various execution sequence lengths is obtained for both the sparse tree representation and the full tree representation.

**Lemma 4.35.** *Let $\sigma$ be a jointed node and let $\sigma_1$ and $\sigma_2$ be jointees of $\sigma$. Then an epistemic calling condition $\varphi$ is satisfied in $\sigma_1$ if and only if $\varphi$ is satisfied in $\sigma_2$.*

*Proof.* Take two arbitrary agents $a_i \neq a_j \in \mathsf{Ag}$. Let the epistemic calling condition for agent $a_i$ to call agent $a_j$ be the formula $K_{a_i}\varphi$. From Definition 4.11, $K_{a_i}\varphi$ is satisfied at a node $\sigma'$ if $\varphi$ is satisfied at every node $\sigma'' \in Z_{a_i}(\sigma')$. But from the definition of equivalence relation given in Definition 4.7, $\sigma_1 \equiv_{a_i} \sigma_2$. Therefore $Z_{a_i}(\sigma_1) = Z_{a_i}(\sigma_2)$. And therefore $K_{a_i}\varphi$ is satisfied in $\sigma_1$ if and only if $K_{a_i}\varphi$ is satisfied in $\sigma_2$. □

**Lemma 4.36.** *Let $\sigma$ be a jointed node and let $\sigma_1$ and $\sigma_2$ be jointees of $\sigma$. Then an epistemic calling condition $\varphi$ is satisfied in $\sigma_1$ if and only if $\varphi$ is satisfied in $\sigma$.*

*Proof.* The proof follows directly from Lemma 4.35: a jointed node is *equal* to one of its jointees. □

**Proposition 4.37.** *The size of the standard extension obtained from the sparse tree is equal to the size of the standard extension obtained from the full tree.*

*Proof.* Let $\sigma_1$ and $\sigma_2$ be child nodes of the root node of a gossip tree $\mathcal{T}_g$. Also let $\sigma_1$ and $\sigma_2$ be jointees. Recall that the value of the multiplicity of the root node is one. Consider the subtree $\tau_1$ which has $\sigma_1$ as its root node, and the subtree $\tau_2$ which has $\sigma_2$ as its root node. From Lemma 4.35 and from Definition 4.7, it follows that $\tau_1$ is identical to $\tau_2$. Therefore if we join $\sigma_1$ and $\sigma_2$ into a node $\sigma$, then the number of paths in the subtree $\tau$ which has the jointed node $\sigma$ as its root will be the number of paths in $\sigma_1$ multiplied by two. Thus the value of the multiplicity of $\sigma$ is two. Inductively, assume that the value of the multiplicity of an arbitrary node $\hat{\sigma}'$ in $\mathcal{T}_g$ is $x$. Suppose $\hat{\sigma}$ is a child node of $\hat{\sigma}'$, then the value of the multiplicity of $\hat{\sigma}$ is $2x$ if $\hat{\sigma}$ is jointed, otherwise the value of the multiplicity of $\hat{\sigma}$ is $x$. We therefore conclude that adding the value of the multiplicity of all the leaf nodes in a sparse tree yields the same number of paths, and thus the same standard extension size, as in the full tree, where such path starts from the root node and terminates in a leaf node of $\mathcal{T}_g$. □

### 4.3.2 Encoding the Agents and the Secrets

Given a scenario with $n$ agents, the set $\mathsf{Ag}$ of agents is encoded as a set of consecutive natural numbers beginning with zero. That is, $\mathsf{Ag} = \{0, 1, \ldots, n-1\}$. The set of secrets known by each agent $i \in \mathsf{Ag}$ is encoded as an $n$-sized binary word, $\omega_i$. Starting from the least significant bit of $\omega_i$, let the $k^{th}$ bit in $\omega_i$ be $\omega_i(k)$, then the $(k+1)^{th}$ bit of $\omega_i$ is set to 1 if and only if agent $i$ knows the secret of agent $k$. Therefore, at the initial gossip

situation where each agent knows only its unique secure, we have that:

$$\begin{aligned} \omega_i(i+1) &= 1 \quad \text{for all} \quad 0 \leq i \leq n-1 \\ \omega_i(j+1) &= 0 \quad \text{for all} \quad 0 \leq i \neq j \leq n-1 \end{aligned}$$

For example, in a scenario with four agents, we have $\mathsf{Ag} = \{0, 1, 2, 3\}$ and the set of secrets known by the agents $0, 1, 2$ and $3$ at the initial gossip situation are, respectively, encoded as $2^0 = 0001$, $2^1 = 0010$, $2^2 = 0100$ and $2^3 = 1000$.

Furthermore, whenever an agent $i$ learns a new secret, say the secret of agent $j$, we set $\omega_i(j+1)$ to 1. So for an arbitrary round $r$ of calls, and an arbitrary pair of agents $i, j \in \mathsf{Ag}$, let the encoding for the secrets known by agent $i$ and $j$ at the end of round $r$ be $\omega_i'$ and $\omega_j'$ respectively. Observe that after a call between $i$ and $j$ at round $r+1$, then the sets $\omega_i''$ and $\omega_j''$ of secrets known by $i$ and $j$, respectively, can be (and are) encoded as:

$$\omega_i'' = \omega_j'' = \omega_i' \oplus \omega_j'$$

Subsequently, to check if an agent $i$ knows the secrets of agent $j$, we test if $\omega_i(j+1) = 1$. The number of secrets known by agent $i$ is given by the number of bits in $\omega_i$ that have the value of 1. Finally, an agent knows all the secrets in the scenario if and only if $\omega_i = 2^n - 1$.

### 4.3.3 Multithreading

To implement the *ComputeNextLayer* procedure, we create $\mu$ threads (lightweight processes), $\mu \geq 1$, and then create as many parallel *tasks* as the number of tree nodes $h'$ in $\lambda_k$. A task generates all the successor nodes $h$ at $h'$. We place each of the newly created tasks in a *task pool*, from where the threads take tasks to execute in parallel. The list of successor nodes is returned by each thread, and all such lists are merged to produce the nodes in the next layer $\lambda_{k+1}$ of the gossip tree. Program execution stops after $n(n-1)/2$ rounds of calls. This corresponds to the maximum number of rounds needed to attain the goal state if there is no redundant call in an execution sequence, that is, in each call some agent learns some new secret (see Proposition 6.13 in Chapter 6).

We utilise the Java Executor service which provides life-cycle routines for dynamic threads, and handles the background management of such threads and their associated task pool. The Java Executor service also provides a mechanism called *Future*: for each new parallel task allocated to a dynamic thread, the executor framework returns a Future object which provides a handle to any result returned as a result of execution of the task. We could then go through the list of all Futures when all the tasks are completed to process their results. For optimal scheduling of the threads, careful consideration of the available CPUs and dynamic memory is important vis-*à*-vis number of created threads.

## 4.4 Related Work

The gossip tree and the EGP tool implement the epistemic gossip protocols described in Chapter 3. The EGP tool can be seen as a dedicated model checker, based on the standard procedures for CTL model checking [33]. The technique of combining the *TopDownSAT* and *BottomUpSAT* procedures is similar to the Bounded Model Checking technique described in [52], which mitigates the state explosion problem by narrowing down the set of situations to the relevant set, that is, the set of those situations that are required to check the satisfiability of a formula at a given situation. We further optimise the computation of the relevant set by introducing a gossip tree layer labelling, which in effect maintains a cache of the equivalence class information from previous rounds.

Other, general purpose, model checkers for epistemic scenarios include DEMO [68], MCK [25] and MCMAS [42]. DEMO allows a user to describe the epistemic properties of a scenario using formulas of dynamic epistemic logic, and a Kripke model for the described scenario is then generated. Epistemic actions are likewise described in action model logic, and DEMO generates an action model for such an action, and then generates the resulting Kripke model due to the execution of the action at an initial scenario. DEMO then enables a user to check the epistemic properties of the resulting model. DEMO also supports a graphical display of epistemic models and action models. Similar to the EGP tool, MCK allows for the explicit specification of the protocol for each agent, and the automated execution of such a protocol in a multiagent setting involving non-deterministic selection of actions. But unlike the EGP tool, MCK supports temporal (LTL and CTL) and epistemic logic specifications. Also, like MCK, the EGP tool provides a custom modelling language, namely EGPL, which can flexibly be adapted for describing and modelling other protocols that are similar to, or are based on, epistemic gossip protocols. Furthermore, MCMAS, similar to MCK, supports model checking of epistemic logic and temporal (CTL) logic specifications. But additionally, MCMAS supports the model checking of alternating time logic specifications [2] and deontic specifications [43].

In principle, our protocols could also be implemented with such tools as DEMO, MCK and MCMAS. For this investigation however we focused on a dedicated tool for epistemic gossip protocols. We leave for a future work the comparative study of the computational performance of these tools on epistemic-gossip-based protocols.

## 4.5 Conclusion

In this chapter we presented an end-to-end description of the tool EGP — a software framework for epistemic gossip protocols. We introduced EGPL — a high level language for describing epistemic gossip protocols, and we described the design of EGPL interpreter which translates a given protocol description into a gossip tree on which the protocol is evaluated. In the next chapter we present experiments and results with the EGP tool, including consideration of network topologies for agent interaction.

Some interesting directions for future versions of the EGP tool are toward capabilities for parallel calls among the agents in the scenario. In addition, the current version of the EGP tool assumes that the calls are made in the private synchronous mode, that is 0-mode calls. In a future version of the EGP tool, it is possible to incorporate the capability of making public synchronous mode (−-mode) calls and the asynchronous mode (+-mode) calls. And this can be realised in a modular manner. We point to a possible direction as follows. The equivalence notion defined in Definition 4.7 is for 0-mode calls. Similar definitions for the −-mode and +-mode calls is given below (see also Definitions 4.5, 4.6 and 4.7).

**Definition 4.38** (Equivalence Relation, $\equiv^{-}$)**.** Let $F(h; a_i a_j)$ and $F(h'; a_k a_l)$ be as in Definition 4.7. Let the call mode be −-mode. We inductively define an equivalence relation between histories as follows: $e \equiv^{-}_{a_m} e$, and $h; a_i a_j \equiv^{-}_{a_m} h'; a_k a_l$ for all $a_m \in \mathsf{Ag}$.

**Definition 4.39** (Equivalence Relation, $\equiv^{+}$)**.** Let $F(h; a_i a_j)$ and $F(h'; a_k a_l)$ be as in Definition 4.7. Let the call mode be +-mode. We inductively define an equivalence relation between histories as follows: $e \equiv^{+}_{a_m} e$, and $h; \beta \equiv^{+}_{a_m} h'; \beta'$ iff: $S_m = S'_m$, and $h \equiv^{+}_{a_m} h'$, where $\beta, \beta' \in \{a_i a_j, a_k a_l, \mathsf{skip}\}$ and $[a_m \in \mathsf{Set}(\beta) = \mathsf{Set}(\beta')$ or $a_m \notin \mathsf{Set}(\beta) \cup \mathsf{Set}(\beta')]$, for all $a_m \in \mathsf{Ag}$, and where $\mathsf{Set}(a_i a_j) = \{a_i, a_j\}$ and $\mathsf{Set}(\mathsf{skip}) = \emptyset$ for all $a_i, a_j \in \mathsf{Ag}$.

In the −-mode, the equivalence relation is an identity. This is obvious since in the −-mode every agent knows whom is calling who at every round of call. As such, in the −-mode the agents do not have any ambiguity as to which is the actual history of calls. In the +-mode, there is the possibility of the skip action, so two equivalent histories for an agent can have different lengths. For the gossip tree structure, the width of the tree is one, in the −-mode. However, in the +-mode, the nodes in a given gossip tree layer $\lambda_k$ can be determined by referring to the gossip tree $\mathcal{T}_g$ given by 0-mode calls, and then basically taking the union of the set of nodes in the layers $1 \leq k$ of $\mathcal{T}_g$.

Finally, consider the following two histories or execution sequences for a scenario with five agents $a, b, c, d, e$:

$$\begin{aligned} h_1 &= cd; da; cd; de; eb \\ h_2 &= ca; cd; ba; de; eb \end{aligned}$$

According to Definition 4.7, $h_1 \equiv_e h_2$, since at every same round in both $h_1$ and $h_2$, either $e$ called with the same other agent or it was not involved in the call, moreover agent $e$ knows the same secrets at the end of each round of call in both execution sequences. But we can also consider that at the fifth call of $h_2$, agent $e$ will learn that agent $b$ had called previously in the execution sequence, since in that fifth round call agent $e$ will learn that agent $b$ already knows secrets $A, B, C$, whereas at the same round in $h_1$ agent $e$ will learn that agent $b$ knows only the secret $B$, which then leads agent $e$ to know that agent $b$ had not called previously in the execution sequence $h_1$. Therefore, if we consider the secrets that agent $e$ learns from its calling partner in each call, then agent $e$ can distinguish between the execution sequences $h_1$ and $h_2$. However, to enrich the agents

with this capability we need to redefine the equivalence relation $\equiv$ given in Definition 4.7. We give an alternative definition as follows:

**Definition 4.40** (Equivalence Relation, $\equiv^{\dagger}$)**.** Let:

$$\begin{array}{lcll} F(h) & = & \langle S_1, \ldots, S_n \rangle & \\ F(h') & = & \langle S'_1, \ldots, S'_n \rangle & \\ F(h; a_i a_j) & = & \langle \dot{S}_1, \ldots, \dot{S}_n \rangle & \text{and} \\ F(h'; a_k a_l) & = & \langle S''_1, \ldots, S''_n \rangle & \end{array}$$

Let the call mode be 0-mode. We inductively define an equivalence relation between histories as follows, for all $a_m \in \mathsf{Ag}$:

**Base Case:**

- $e \equiv^{\dagger}_{a_m} e$

**Inductive Case:**

$h; a_i a_j \equiv^{\dagger}_{a_m} h'; a_k a_l$ iff:

- $\dot{S}_m = S''_m$, and $h \equiv^{\dagger}_{a_m} h'$, and
- $[a_m \in \{a_i, a_j\} = \{a_k, a_l\}$ and $S_x = S'_x$, where $x \in \{a_i, a_j\} \setminus \{a_m\}]$ or $[a_m \notin \{a_i, a_j\} \cup \{a_k, a_l\}]$.

That is, an agent considers two histories to be equivalent if it calls the same other agent in each corresponding call of both histories (or, if it was not involved in both corresponding calls), learns the same secrets from that other agent in the corresponding calls of both histories, and knows the same secrets at the end of each corresponding call in both histories.

However, from the foregoing example execution sequences $h_1$ and $h_2$, observe that the equivalence relation given by Definition 4.40 can lead to an increased extension size of a given protocol since agent $e$ can now distinguish both histories by means of Definition 4.40 (see Corollary 4.43).

**Lemma 4.41.** *Given any epistemic gossip protocol* $\Pi$*, let* $\equiv$ *be the equivalence relation given in Definition 4.7 and let* $\equiv^{\dagger}$ *be the equivalence relation given in Definition 4.40. Let* $\mathcal{T}_g = \langle H, R, F, \{Z_{a_m}\} \rangle$ *and* $\mathcal{T}^{\dagger}_g = \langle H^{\dagger}, R^{\dagger}, F^{\dagger}, \{Z^{\dagger}_{a_m}\} \rangle$ *be gossip trees for* $\Pi$*, where, for any* $a_m \in \mathsf{Ag}$*, we have that* $Z_{a_m}$ *is as given in Definition 4.9 (that is, for any* $\sigma \in H$*,* $\sigma \equiv_{a_m} \sigma'$ *for all* $\sigma' \in Z_{a_m}(\sigma)$*), and where* $Z^{\dagger}_{a_m}$ *is such that for any* $\bar{\sigma} \in H^{\dagger}$*,* $\bar{\sigma} \equiv^{\dagger}_{a_m} \sigma'$ *for all* $\sigma' \in Z^{\dagger}_{a_m}(\bar{\sigma})$*. Then, for any* $\sigma$ *such that* $\sigma \in H$ *and* $\sigma \in H^{\dagger}$*,*

$$Z^{\dagger}_{a_m}(\sigma) \subseteq Z_{a_m}(\sigma)$$

*Proof.* Let $\sigma = \sigma_1, \ldots, \sigma_n$ and $\sigma' = \sigma'_1, \ldots, \sigma'_n$ be any two execution sequences or histories. From Definition 4.7 and Definition 4.40, observe that the set of conditions for the equivalence of $\sigma$ and $\sigma'$ for an agent, are the same for $\equiv$ and $\equiv^{\dagger}$, except that the set of

conditions under $\equiv^{\dagger}$ includes a further condition that an agent must learn the same set of secrets in each corresponding pair of calls $\sigma_i$ in $\sigma$, and $\sigma_j$ in $\sigma'$, for all $i = j$. Therefore, beginning from the empty history $e$, and following the inductive definitions in Definition 4.7 and Definition 4.40, we obtain that $\sigma \equiv^{\dagger} \sigma'$ implies $\sigma \equiv \sigma'$, for any pair of histories $\sigma, \sigma' \in H^{\dagger}$. Therefore we conclude that $Z^{\dagger}_{a_m}(\sigma) \subseteq Z_{a_m}(\sigma)$. □

**Proposition 4.42.** *Given any epistemic gossip protocol* $\Pi$*, let* $\equiv$ *be the equivalence relation given in Definition 4.7 and let* $\equiv^{\dagger}$ *be the equivalence relation given in Definition 4.40. Let* $\mathcal{T}_g = \langle H, R, F, \{Z_{a_m}\}\rangle$ *and* $\mathcal{T}^{\dagger}_g = \langle H^{\dagger}, R^{\dagger}, F^{\dagger}, \{Z^{\dagger}_{a_m}\}\rangle$ *be gossip trees for* $\Pi$*, where, for any* $a_m \in \mathsf{Ag}$*, we have that* $Z_{a_m}$ *is as given in Definition 4.9 (that is, for any* $\sigma \in H$*,* $\sigma \equiv_{a_m} \sigma'$ *for all* $\sigma' \in Z_{a_m}(\sigma)$*), and where* $Z^{\dagger}_{a_m}$ *is such that for any* $\bar{\sigma} \in H^{\dagger}$*,* $\bar{\sigma} \equiv^{\dagger}_{a_m} \sigma'$ *for all* $\sigma' \in Z^{\dagger}_{a_m}(\bar{\sigma})$*. Then,*

$$H \subseteq H^{\dagger}$$

From Proposition 4.42 we learn that if a history or an execution sequence $\sigma$ of calls can be made based on the equivalence notion given in Definition 4.7, then that same execution sequence $\sigma$ of calls can be made based on the equivalence notion given in Definition 4.40. That is, any call sequence that is possible under the equivalence notion given in Definition 4.7, is also possible under the equivalence notion given in Definition 4.40, for any epistemic gossip protocol. But on the other hand, some calls that are possible under the equivalence notion given in Definition 4.40, are not possible under the equivalence notion given in Definition 4.7.

*Proof of Proposition 4.42.* Choose an arbitrary pair of agents, $a_i \neq a_j \in \mathsf{Ag}$, and let the epistemic calling condition for agent $a_i$ to call agent $a_j$ for protocol $\Pi$ be $\varphi_{\Pi}(a_i, a_j)$. For any history or execution sequence $\sigma$, let $l(\sigma)$ be the length of $\sigma$, where $l(e) = 0$ and $l(\sigma; a_i a_j) = l(\sigma) + 1$. We proceed by means of an inductive argument, as follows.

**Base Case:** For the initial gossip situation, we have that $e \in H$ and $e \in H^{\dagger}$ (given in Definition 4.7 and Definition 4.40, respectively). Therefore we obtain that $e \in H$ implies $e \in H^{\dagger}$. So when only the initial gossip situation is considered, we have that $H \subseteq H^{\dagger}$.

**Inductive Hypothesis:** Let $\sigma$ be an arbitrary execution sequence in $H$. Suppose that $\sigma' \in H$ implies $\sigma' \in H^{\dagger}$, for every execution sequence $\sigma'$ such that $l(\sigma') \leq l(\sigma)$. Now, based on the definition of epistemic calling condition given in Definition 3.23, and from the semantics of '$K_{a_i}$' operator on a gossip tree (see Definition 4.11 and

Definition 4.16), we have that:

$$\begin{aligned}
\mathcal{T}_g, \sigma \models \varphi_\Pi(a_i, a_j) &\iff \mathcal{T}_g, \sigma \models K_{a_i}\psi(a_i, a_j) && \text{(from Definition 3.23)} \\
&\iff \mathcal{T}_g, \sigma \models K_{a_i}\varphi_\Pi(a_i, a_j) \\
&\qquad \text{(positive introspection, negative introspection)} \\
&\iff \mathcal{T}_g, \sigma' \models \varphi_\Pi(a_i, a_j) \text{ for all } \sigma' \in Z_{a_i}(\sigma) \\
&\qquad \text{(Definition 4.11: semantics of '}K_{a_i}\text{' operator)} \\
&\implies \mathcal{T}_g^\dagger, \sigma' \models \varphi_\Pi(a_i, a_j) \text{ for all } \sigma' \in Z_{a_i}^\dagger(\sigma) \\
&\qquad \text{(from the inductive hypothesis and Lemma 4.41)} \\
&\iff \mathcal{T}_g^\dagger, \sigma \models \varphi_\Pi(a_i, a_j) \\
&\qquad \text{(Definition 4.11: semantics of '}K_{a_i}\text{' operator)}
\end{aligned} \tag{‡}$$

From (‡), we see that $\sigma; a_i a_j \in H$ implies $\sigma'; a_i a_j \in H^\dagger$. And since we considered an arbitrary $\sigma \in H$, and an arbitrary pair of agents $a_i \neq a_j \in \mathsf{Ag}$, we therefore conclude that $H \subseteq H^\dagger$. □

**Corollary 4.43.** *Given any epistemic gossip protocol* $\Pi$*, let* $\Sigma(\Pi)$ *be the extension of* $\Pi$ *obtained by assuming the notion of equivalence of histories given in Definition 4.7, and let* $\Sigma^\dagger(\Pi)$ *be the extension of* $\Pi$ *obtained by assuming the notion of equivalence of histories given in Definition 4.40. Then* $\Sigma(\Pi) \subseteq \Sigma^\dagger(\Pi)$.

*Proof.* The corollary follows from Proposition 4.42 since $H$ is equal to $\Sigma(\Pi)$, and $H^\dagger$ is equal to $\Sigma^\dagger(\Pi)$ (See Definition 4.16). □

We refer the reader to Section 5.4 for a further comparison of Definitions 4.7 and 4.40 based on empirical data.

# Chapter 5

# Experiments and Results

## 5.1 Introduction

In the previous chapter we introduced the Epistemic Gossip Protocol tool (EGP), which enables protocol designers to specify epistemic gossip protocols in terms of the Epistemic Gossip Protocol Programming Language (EGPL). Given such a specification, the EGPL interpreter generates the gossip tree corresponding to the specified protocol, and gives key dynamic properties of the protocol.

Gossip protocols are based on communication networks which serve as the environment in which agents interact and share their secrets. In Chapter 3 we assumed that the underlying network of agents is a complete topology network. That is, for each pair of distinct agents there is a direct communication link between the agents of such pair. As such, if the epistemic calling condition for the given protocol holds for a pair of agents, then the agents of such pair can communicate with each other by means of a call and exchange all the secrets they know.

Although we assume gossip scenarios in which the underlying network graph is strongly connected (hence the physical possibility that every agent learns all the secrets through gossiping), it is also the case that many communication networks form incomplete graphs. Therefore, in addition to studying the performance of epistemic gossip protocols on complete topology networks, it is desirable to study the behaviour of such protocols on various other network topologies. For this purpose the EGPL language also enables the description of the underlying network graph in a gossip scenario. We will exemplify the use of this feature by describing the various network topologies given in Section 2.3, for use with our epistemic gossip protocols.

For this chapter, we now proceed to describe our experimental setup. We also discuss how we interpret the empirical results we obtain by analysing epistemic gossip protocols using the EGP tool. In subsequent sections, and for complete, line, star, binary tree and circle topology networks, we present EGPL descriptions and empirical results for Learn New Secrets (Protocol 1), Known Information Growth de Dicto (Protocol 2), Known Information Growth de Re (Protocol 3), Possible Information Growth de Dicto (Protocol 4) and Possible Information Growth de Re (Protocol 5). In the next chapter

we will present theoretical results that describe further properties of the aforementioned protocols on the various network topologies considered in this chapter.

## 5.2 Experimental Setup

The aim of our experiments is to evaluate and compare various epistemic gossip protocols with respect to their performance on the gossip problem. We examine the performance of five protocols based on three properties, (i) time and space efficiency, (ii) adaptability, and (iii) scalability. These properties are the typical performance measures of multiagent systems in the literature [48]. In the gossip protocol literature it is typical to measure the performance of a protocol by considering the length of its execution sequence, that is, the number of calls in the execution sequence [30]. Similarly we measure the performance of epistemic gossip protocols in terms of execution sequence length, together with the size of the standard extension* of the given protocol. Given the epistemic calling condition of a terminating epistemic gossip protocol and the network topology of the underlying network of agents, we define the time efficiency of an epistemic gossip protocol as: the average length of all the execution sequences in the protocol's standard extension as a function of the number of agents in the scenario. Likewise, we define the space efficiency of an epistemic gossip protocol as: the size of the standard extension of the protocol as a function of the number of agents in the scenario. Since the agents reason about the possible execution sequences contained in the extension of the protocol (or, alternatively, the agents reason about the prefixes of the possible execution sequences in the standard extension of the protocol), therefore the standard extension size gives an indication of the computing memory required by the agents in the scenario for the given protocol.

The scalability property measures the performance of the protocol with increase in the size of the gossip scenario, that is, by increasing the number of agents and, correspondingly, secrets in the scenario. Upon increasing the number of agents in the gossip scenario, we observe the relative change in the time and space efficiency of two given protocols, and the protocol whose efficiency is less negatively impacted by the increase is considered more scalable. With the adaptability property we measure the performance of the protocol under changes or disruption in the physical arrangement of the network of agents in the system. Specifically we study the adaptability of epistemic gossip protocols by measuring their time and space efficiencies on various network topologies.

We also present empirical analysis of the termination and successfulness properties of the protocols. Recall that a protocol is *terminating* if all its execution sequences are finite, otherwise it is *non-terminating*; and an execution sequence of a gossip protocol is *successful* if and only if it is terminal, and gives rise to the goal state when executed at the initial state, where the initial state is that in which each agent knows only its own unique secret and the goal state is that in which each agent knows the secrets of all the other agents (see also Definition 4.19). An epistemic gossip protocol is successful

*See Definition 4.21 for the definition of standard extension

if and only if all its execution sequences are terminating, and all its terminal execution sequences give rise to a goal state (see also Definition 4.20). An execution sequence is *deadlocked* if no further calls can be made in the execution sequence *and* the goal state is not reached by the execution sequence. Note that given our definition of a successful/unsuccessful execution sequence of an epistemic gossip protocol we consider both a deadlocked sequence and an infinitely looping execution sequence to be equally unsuccessful. This approach serves us well as a common criteria by which to compare protocols whose extension contain (i) deadlocked sequences, or (ii) infinitely looping sequences, or (iii) both deadlocked and infinitely looping sequences. The comparison is then based on a percentage of successful/unsuccessful sequences, which we will see as part of the table of values obtained from the EGP tool for the characteristics of the various epistemic gossip protocols we analyse.

Finally, all our measurements are made from the EGP tool after executing a given epistemic gossip protocol for $r_{max} = n(n-1)/2$ rounds, where $r_{max}$ is the maximum number of calls needed in an execution sequence of the protocol to reach the goal state, assuming that only a pair of agents call each other in each call, and that some secret is learnt by at least one of the calling pair in each call (that is to say, the epistemic gossip protocol is regular - see Definition 6.3; also see Proposition 6.13). We chose our value of $r_{max}$ for our experiments because, apart from Possible Information Growth de Re and Possible Information Growth de Dicto protocols which are non-terminating, the rest of our example protocols are regular protocols (see Proposition 6.9).

## 5.3 Protocol Descriptions and Results

In this section we present the EGPL description of our example gossip protocols on the network topologies described in Section 2.3. Each protocol description consists of an epistemic calling condition whose formal language is the language of $\mathcal{L}_{cc}$ defined in the previous chapter. The epistemic calling condition is then followed by an optional network topology description. The default network topology is the complete topology network, which is assumed if no network topology is specified in the EGPL description. In the subsections that follow we present the EGPL description for the complete, line, circle, star and binary tree topology networks. We then present performance results obtained by interpreting the EGPL description of the protocols described.

### 5.3.1 Protocol Descriptions on Complete Topology Network

Listings 5.1, 5.2 and 5.3 are EGPL descriptions of Protocols 1, 2 and 3, respectively. The specification of the network topology is omitted, therefore the complete topology network will be assumed. That is, the EGPL interpreter will assume that every pair of agents in the scenario has a direct physical communication link between them. So any pair of agents in the scenario will be able to communicate with each other if the epistemic

calling condition is satisfied for the pair [†]. In Listing 5.1, the epistemic calling condition for Protocol 1 is described in lines 3-5. Particularly, the calling condition expressed in line 4 is equivalent to the epistemic calling condition for Protocol 1, which is reproduced here as the formula $\varphi_{\Pi_1}(a_i, a_j)$ in Equation (5.1), where $\varphi_{\Pi_1}(a_i, a_j) \in \mathcal{L}_{cc}$.

$$\varphi_{\Pi_1}(a_i, a_j) = K_{a_i} \neg K w_{a_i} A_j \tag{5.1}$$

```
1  begin
2       /* epistemic calling condition */
3       let ai call aj if {
4         ai knows (init(aj) \notin secret(ai));
5       }
6  end
```

LISTING 5.1: **EGPL Description for Protocol 1 on Complete Topology Network**

In Listing 5.2, the epistemic calling condition for Protocol 2 is described in lines 2-4. Again, in the description, `ai` and `aj` are agent name variables. Particularly, the epistemic calling condition expressed in line 3 is equivalent to the epistemic calling condition for Protocol 2, which is reproduced here as the formula $\varphi_{\Pi_2}(a_i, a_j)$ in Equation (5.2), where $\varphi_{\Pi_2}(a_i, a_j) \in \mathcal{L}_{cc}$ (note that $\mathsf{P}$ is the set of all secrets in the scenario).

$$\varphi_{\Pi_2}(a_i, a_j) = K_{a_i}((\bigvee_{A_k \in \mathsf{P}} (K w_{a_i} A_k \wedge \neg K w_{a_j} A_k)) \vee (\bigvee_{A_k \in \mathsf{P}} (\neg K w_{a_i} A_k \wedge K w_{a_j} A_k))) \tag{5.2}$$

The formula $\varphi_{\Pi_2}(a_i, a_j)$ says that $a_i$ knows that there is some secret $A_k$ which only one of $a_i$ and $a_j$ knows. The language EGPL allows us to express this epistemic calling condition succinctly as shown in Listing 5.2, line 3.

```
1  begin
2       let ai call aj if {
3         ai knows (secret(ai) != secret(aj));
4       }
5  end
```

LISTING 5.2: **EGPL Description for Protocol 2 on Complete Topology Network**

In Listing 5.3, the epistemic calling condition for Protocol 3 is described in lines 2-7. Particularly, the epistemic calling condition expressed in lines 3-6 is equivalent to the formula $\varphi_{\Pi_3}(a_i, a_j)$ as given in Equation (5.3), where $\varphi_{\Pi_3}(a_i, a_j) \in \mathcal{L}_{cc}$.

$$\varphi_{\Pi_3}(a_i, a_j) = (\bigvee_{A_k \in \mathsf{P}} K_{a_i}((K w_{a_i} A_k \wedge \neg K w_{a_j} A_k) \vee (\neg K w_{a_i} A_k \wedge K w_{a_j} A_k))) \tag{5.3}$$

[†]In all the EGPL descriptions given in this chapter, `ai`, `aj`, `ak`,... are agent name variables. The EGPL interpreter will substitute real and unique agent names for the name variables to obtain an actual epistemic calling condition.

```
1  begin
2       let ai call aj if {
3         disjunct ak: {ai knows (
4           (init(ak) \in (secret(ai) \cup secret(aj))) &&
5           (init(ak) \notin (secret(ai) \cap secret(aj)))
6         )};
7       }
8  end
```

LISTING 5.3: **EGPL Description for Protocol 3 on Complete Topology Network**

The formula $\varphi_{\Pi_3}(a_i, a_j)$ says that there is some secret $A_k$ such that $a_i$ knows that $A_k$ is known by only one of agent $a_i$ and $a_j$.

Similar to Listing 5.2 and 5.3, Listings 5.4 and 5.5 describe the epistemic calling condition of Protocol 4 and 5, respectively. The epistemic calling condition for Protocol 4 is described in lines 2-4 of Listing 5.4. Particularly, the calling condition expressed in line 3 is equivalent to the formula $\varphi_{\Pi_4}(a_i, a_j)$ in Equation (5.4), where $\varphi_{\Pi_4}(a_i, a_j) \in \mathcal{L}_{cc}$.

$$\varphi_{\Pi_4}(a_i, a_j) = \neg K_{a_i} \neg (( \bigvee_{A_k \in \mathsf{P}} (K w_{a_i} A_k \wedge \neg K w_{a_j} A_k)) \vee ( \bigvee_{A_k \in \mathsf{P}} (\neg K w_{a_i} A_k \wedge K w_{a_j} A_k))) \quad (5.4)$$

```
1  begin
2       let ai call aj if {
3         ai \neg knows \neg (secret(ai) != secret(aj));
4       }
5  end
```

LISTING 5.4: **EGPL Description for Protocol 4 on Complete Topology Network**

The epistemic calling condition for Protocol 5 is described in lines 2-7 of Listing 5.5. The epistemic calling condition is expressed in lines 3-6, and it is equivalent to the formula $\varphi_{\Pi_5}(a_i, a_j)$ in Equation (5.5), where $\varphi_{\Pi_5}(a_i, a_j) \in \mathcal{L}_{cc}$.

$$\varphi_{\Pi_5}(a_i, a_j) = ( \bigvee_{A_k \in \mathsf{P}} \neg K_{a_i} \neg ((K w_{a_i} A_k \wedge \neg K w_{a_j} A_k) \vee (\neg K w_{a_i} A_k \wedge K w_{a_j} A_k))) \quad (5.5)$$

```
1  begin
2       let ai call aj if {
3         disjunct ak: {ai \neg knows \neg (
4           (init(ak) \in (secret(ai) \cup secret(aj))) &&
5           (init(ak) \notin (secret(ai) \cap secret(aj)))
6         )};
7       }
8  end
```

LISTING 5.5: **EGPL Description for Protocol 5 on Complete Topology Network**

TABLE 5.1: **Protocol 1 on Complete Topology Network.**

| Execution Sequence Length | Three Agents | Four Agents | Five Agents |
|---|---|---|---|
| 3 | 24 | | |
| 4 | | 384 | |
| 5 | | 2,496 | |
| 6 | | 2,688 | 103,680 |
| 7 | | | 1,614,720 |
| 8 | | | 5,285,760 |
| 9 | | | 6,913,920 |
| 10 | | | 3,492,480 |
| Standard Extension Size | 24 | 5,568 | 17,410,560 |
| Average Execution Sequence Length | 3 | 5.41379 | 8.69365 |
| Successful Sequences | 24 | 5,568 | 17,410,560 |
| % Successful Sequences | 100.00% | 100.00% | 100.00% |

### 5.3.2 Protocol Results on Complete Topology Network

In Tables 5.1-5.5 we present the protocol characteristics obtained from the EGP tool for Protocols 1 through 5, respectively. The tables show the number of length $x$ sequences for $3, 4, 5$ agents, where $3 \leq x \leq 10$. We also show the standard extension size and average execution sequence length for the protocols.

**Performance Analysis.** Our empirical results for scenarios with 3, 4 and 5 agents show that Protocols 1, 2 and 3 are all successful after $n(n-1)/2$ rounds. From the numerical results for average execution sequence length and standard extension size, respectively, we observe that Protocol 1 is more time and space efficient than Protocol 2 and Protocol 3. We also note that Protocol 3 proves significantly more space efficient and slightly more time efficient than Protocol 2.

As shown in the proof of Proposition 3.29, Protocols 4 and 5 are non-terminating because there is the possibility of infinitely looping execution sequences in these protocols. But we see that for the scenario comprising of three agents, Protocol 4 and Protocol 5 terminate and are 100% successful. For the scenario comprising of four agents we have up to 94.48% successful sequences, although the resulting standard extension size for each of both protocols in the four-agent scenario is about five times the standard extension size of the corresponding scenario size for Protocol 3, about four times the standard extension size of the corresponding scenario size for Protocol 2 and about sixty-three times that of the corresponding scenario size for Protocol 1. Note that in accordance with the theory of Chapter 3, every successful sequence in Protocol 4 is also in Protocol 5, and vice versa (the number of successful sequences for the three- and four-agent scenarios of Protocols 4 and 5 are the same).

To measure the scalability of the protocols on a complete topology network we consider the percentage increase in the average execution sequence length of the protocols as a result of increasing the size of the gossip scenario by one (that is, adding one more agent to the scenario, and correspondingly, one more secret). Let the average execution sequence length of a protocol for a scenario consisting of $n$ agents be $\alpha_n$, then the percentage increase $\gamma$ of the average execution sequence length as a result of introducing one more agent is given by:

$$\gamma_{n,n+1} = \frac{\alpha_{n+1} - \alpha_n}{\alpha_n}$$

TABLE 5.2: **Protocol 2 on Complete Topology Network.**

| Execution Sequence Length | Three Agents | Four Agents | Five Agents |
|---|---|---|---|
| 3 | 96 | | |
| 4 | | 384 | |
| 5 | | 15,744 | |
| 6 | | 64,896 | 195,840 |
| 7 | | | 7,958,400 |
| 8 | | | 61,155,840 |
| 9 | | | 220,404,480 |
| 10 | | | 472,988,160 |
| Standard Extension Size | 96 | 81,024 | 762,702,720 |
| Average Execution Sequence Length | 3 | 5.79621 | 9.51833 |
| Successful Sequences | 96 | 81,024 | 762,702,720 |
| % Successful Sequences | 100.00% | 100.00% | 100.00% |

TABLE 5.3: **Protocol 3 on Complete Topology Network.**

| Execution Sequence Length | Three Agents | Four Agents | Five Agents |
|---|---|---|---|
| 3 | 96 | | |
| 4 | | 384 | |
| 5 | | 13,824 | |
| 6 | | 53,952 | 149,760 |
| 7 | | | 5,798,400 |
| 8 | | | 37,975,680 |
| 9 | | | 172,362,240 |
| 10 | | | 325,891,200 |
| Standard Extension Size | 96 | 68,160 | 542,177,280 |
| Average Execution Sequence Length | 3 | 5.78592 | 9.50882 |
| Successful Sequences | 96 | 68,160 | 542,177,280 |
| % Successful Sequences | 100.00% | 100.00% | 100.00% |

TABLE 5.4: **Protocol 4 on Complete Topology Network.**

| Execution Sequence Length | Three Agents | Four Agents | Five Agents |
|---|---|---|---|
| 3 | 96 | | - |
| 4 | | 384 | - |
| 5 | | 31,488 | - |
| 6 | | 297,984 | - |
| 7 | | | - |
| 8 | | | - |
| 9 | | | - |
| 10 | | | - |
| Standard Extension Size | 96 | 349,134 | - |
| Average Execution Sequence Length | 3 | - | - |
| Successful Sequences | 96 | 329,856 | - |
| % Successful Sequences | 100.00% | 94.48% | - |

TABLE 5.5: **Protocol 5 on Complete Topology Network.**

| Execution Sequence Length | Three Agents | Four Agents | Five Agents |
|---|---|---|---|
| 3 | 96 | | - |
| 4 | | 384 | - |
| 5 | | 31,488 | - |
| 6 | | 297,984 | - |
| 7 | | | - |
| 8 | | | - |
| 9 | | | - |
| 10 | | | - |
| Standard Extension Size | 96 | 349,134 | - |
| Average Execution Sequence Length | 3 | - | - |
| Successful Sequences | 96 | 329,856 | - |
| % Successful Sequences | 100.00% | 94.48% | - |

We summarise $\gamma$ for Protocols 1, 2 and 3 in Table 5.6.

TABLE 5.6: **Protocol scalability on Complete Topology Network.**

| Percentage Increase | Protocol 1 | Protocol 2 | Protocol 3 |
|---|---|---|---|
| $\gamma_{3,4}$ | 80.4597% | 93.2070% | 92.8640% |
| $\gamma_{4,5}$ | 60.5834% | 64.2164% | 64.3441% |

Looking at Table 5.6, we observe that Protocol 1 is the most scalable of the three protocols. However we observe that Protocol 2 shows a greater growth than Protocol 3 when increasing the size of the scenario from three to four agents, but the reverse is the case when increasing the size of the scenario from four to five agents. So it is not yet clear which is more scalable between Protocol 2 and 3. We will also carry out a similar analysis for line, star and binary tree topology networks (we will skip this analysis for the circle topology network because our empirical results show that on a circle topology network most of the protocols are unsuccessful in scenarios with more than three agents). Notice also that we skipped the scalability analysis for Protocols 4 and 5 because they are both non-terminating.

**Extension Analysis.** Let us consider some extension analysis for a scenario that consists of four agents $a, b, c, d$. Example execution sequences of Protocol 1 are $ab; ac; de$; $ad$; $bd; ce$, $ab; ac; de; ad; ce; bc$ and $ab; ac; de; ad; ce; bd$. The execution sequence $ab; ac; bd$; $ad; ab; bc$ is an execution sequence of Protocol 2 but not an execution sequence of Protocol 3. Not all execution sequences of Protocol 4 and Protocol 5 are successful. Some examples of unsuccessful execution sequences of Protocol 5 are: $ab; cd; ab; bc; ab; cd; ab$, $ad; bc; ad; bc; ad; bc; ad$ and $bd; ac; bd; ac; bd; ac; bd$. Examples of successful execution sequences of Protocol 5 are: $cd; ab; bd; ac$, $ab; ac; bd; cd; ab$ and $bc; cd; bc; ab; bd; bc$.

Not every execution sequence of Protocol 2 is an execution sequence of Protocol 1, and not every execution sequence of Protocol 3 is an execution sequence of Protocol 1. For example, the following execution sequences are execution sequences of Protocol 2 but they are not execution sequences of Protocol 1: $ab; ac; ad; ac; ab$, $bd; ad; bd; ac; ab; cd$ and $cd; bd; bc; ab; ac; bd$. Not every successful execution sequence of Protocol 5 is an execution sequence of Protocol 3, for example: $ac; ab; ad; ac; bc$, $bd; cd; ad; bd; bc$ and $ab; ac; bd; ad; ac; bd$. Also, not every successful execution sequence of Protocol 4 is an execution sequence of Protocol 2, examples are: $ad; bd; ac; ad; bc$, $bc; ab; ad; cd; bc$ and $cd; bd; ac; cd; ac; bc$. Examples of execution sequences of Protocol 2 that are not execution sequences of Protocol 3 are: $ab; ac; bd; ad; ab; bc$, $bd; cd; bc; ab; ad; ac$ and $cd; ad; ab; cd$; $ac; bd$.

**Comparison with Literature.** From Tables 5.1-5.5 we see immediately that the minimum successful execution sequence lengths agree with the theoretical result obtained for the minimum successful execution sequence length in the non-epistemic traditional gossip literature for the complete topology network [13, 27, 61, 63]. Furthermore, recall that for a random protocol (that is, wherein a pairwise call is staged at random in each round),

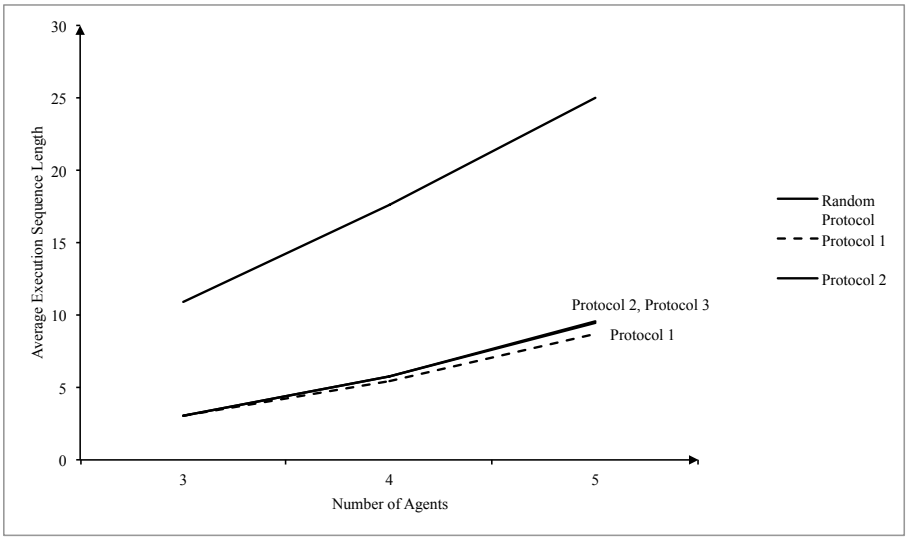



FIGURE 5.1: **Comparing Protocols 1, 2 and 3 to a random protocol.**

Boyd and Steele [11] showed that the average number $A(n)$ of pairwise communicative interactions needed for successful gossiping among $n$ agents is:

$$A(n) = \frac{3}{2} n \; ln \; n + O(n(ln \; n)^{0.5}) \tag{5.6}$$

The graph shown in Figure 5.1 compares the average execution sequence lengths obtained for Protocols $1-3$ with those obtained from Equation 5.6 for a random protocol. Clearly the epistemic gossip protocols (Protocols 1, 2 and 3) outperform the random protocol.

### 5.3.3 Protocol Descriptions on Line Topology Network

Listings 5.6 and 5.7 present EGPL descriptions of Protocols 1 and 2, respectively, on a line topology network. In Listing 5.1, the epistemic calling condition for Protocol 1 is described in lines 2-4, and the network topology is described in lines 5-10 for a network of five agents $a, b, c, d, e$. Note that in the epistemic calling condition (lines 3-5), `ai` and `aj` are agent name variables, in contrast to real agent names used to describe network topology[‡]. As described earlier in the chapter, the agents form the nodes of the network graph, and the neighbourhood relations shown in lines 7-10 of Listing 5.6 express graph node neighbourhood over the agents in the network[§].

```
1   begin
2         let ai call aj if {
3           ai knows (init(aj) \notin secret(ai));
4         }
5         topology {
```

[‡]Although we use agent name variables in the epistemic calling conditions in the EGPL descriptions, we use *real* agent names in the topology description, throughout this chapter

[§]Note that in the EGPL description of the network topology we assume that 'a neighbour b' if and only if 'b neighbour a', where $a$ and $b$ are any given nodes in the network graph

```
6           a neighbour b;
7           b neighbour c;
8           c neighbour d;
9           d neighbour e;
10        }
11  end
```

LISTING 5.6: **EGPL Description for Protocol 1 on Line Topology Network (Five Agents)**

In order to impose a line topology network on other protocols, a similar description to that shown in lines 5-10 of Listing 5.6 is inserted, for the agents in the scenario, after the epistemic calling condition of the protocol is described. In Listing 5.7 we give a line topology network description for a scenario with four agents, and for Protocol 2.

```
1   begin
2         let ai call aj if {
3           ai knows (secret(ai) != secret(aj));
4         }
5         topology {
6           a neighbour b;
7           b neighbour c;
8           c neighbour d;
9         }
10  end
```

LISTING 5.7: **EGPL Description for Protocol 2 on Line Topology Network (Four Agents)**

### 5.3.4 Protocol Results on Line Topology Network

**Performance Analysis.** The table for the performance of Protocol 3 is identical to that for Protocol 2 (shown in Table 5.7). This indicates that these two protocols are in fact identical on a line topology network (for a formal proof about this point, see Corollary 6.39 in Section 6.3, later). Hence from our experiments, both Protocol 2 and Protocol 3 have the same scaling properties on a line topology network. With respect to adaptability property, note that, as shown in Table 5.7, Protocol 2 (and 3) exhibit 100% success for their execution sequences for the scenarios comprising of 3, 4 and 5 agents on a line topology network (same as on a complete topology network). (In Section 6.3 we prove that Protocol 2 and Protocol 3 are successful for all $n = |Ag|$ on a line topology network). Furthermore we observe that for both Protocols 2 and 3 we obtain a better average execution sequence lengths and smaller standard extension sizes on a line topology network than on a complete topology network. Note that although the line topology network induces a lower average execution sequence length than the complete topology network, for Protocol 2 and 3, there are no length four execution sequences for the four-agent scenario, and there are no length six execution sequences for the five-agent scenario. (In Section 6.3 we prove that for $n > 3$ we cannot obtain the shortest successful

execution sequence length of $2n - 4$ on the line topology network for Protocol 2 (and similarly for Protocol 3)).

Finally, the table for the performance of Protocol 5 is identical to that for Protocol 4 (shown in Table 5.8). We observe that the line topology network setting yields a greater percentage of successful execution sequences for these protocols, than in the complete topology network setting.

For the scalability property of the protocols on a line topology network, we observe that the standard extension of both Protocol 2 and Protocol 3 are identical for the experimental cases, therefore the scalability properties of both protocols are the same on this network topology (see the proof of Proposition 6.39 in Chapter 6, where we show that both Protocol 2 and Protocol 3 have identical standard extension for all $n > 0$, where $n$ is the number of agents in the scenario). Note also that Protocol 1 is not successful on this network topology for a scenario with more than two agents, so we skip the scalability analysis for this protocol here.

TABLE 5.7: **Protocol 2 on Line Topology Network.**

| Execution Sequence Length | Three Agents | Four Agents | Five Agents |
|---|---|---|---|
| 3 | 16 | | |
| 4 | | 0 | |
| 5 | | 192 | |
| 6 | | 512 | 0 |
| 7 | | | 2,048 |
| 8 | | | 16,512 |
| 9 | | | 39,424 |
| 10 | | | 59,392 |
| Standard Extension Size | 16 | 704 | 117,376 |
| Average Execution Sequence Length | 3 | 5.72727 | 9.33043 |
| Successful Sequences | 16 | 704 | 117,376 |
| % Successful Sequences | 100.00% | 100.00% | 100.00% |

TABLE 5.8: **Protocol 4 on Line Topology Network.**

| Execution Sequence Length | Three Agents | Four Agents | Five Agents |
|---|---|---|---|
| 3 | 16 | | |
| 4 | | 0 | |
| 5 | | 192 | |
| 6 | | 768 | 0 |
| 7 | | | 2,560 |
| 8 | | | 34,560 |
| 9 | | | 184,576 |
| 10 | | | 610,560 |
| Standard Extension Size | 16 | 974 | 860,248 |
| Average Execution Sequence Length | 3 | - | - |
| Successful Sequences | 16 | 960 | 832,256 |
| % Successful Sequences | 100.00% | 98.56% | 96.75% |

**Extension Analysis.** For Protocol 1, there are no successful execution sequences on a line topology network, for a scenario with more than two agents. This is easy to see: consider a gossip scenario with at least three agents $a, b, c, \ldots$ Assume the network topology shown in Listing 5.6 for these agents. So now agent $b$ is *in between* agent $a$ and agent $c$ in the network, such that, although the epistemic calling condition of Protocol 1 will be satisfied for agent $a$ to call agent $c$ (or for agent $c$ to call agent $a$), but since there is no direct link between agent $a$ and $c$, the only way for *both* agents to know each other's secrets is through the subsequence $ab; \ldots; bc; \ldots; ba$ or $bc; \ldots; ab; \ldots; bc$ (or their reverse

call variants). But such subsequences do not comply with Protocol 1 (since no two agents agents can call each other more than once in an execution sequence of Protocol 1). So in Protocol 1 on a line topology network, one of agents $a$ and $c$, will never know all the secrets in the scenario. (See also Section 6.3).

For the scenario described in Listing 5.7 (Protocol 2), the shortest successful execution sequence is of length five. An example of such a sequence is $ab; cd; bc; ba; cd$. An example of a longest successful execution sequence for the same description is $cd; bc; ab; cd; bc; cd$.

**Comparison with Literature.** From Table 5.7 and Table 5.8 we see that the minimum successful execution sequence lengths agree with the theoretical results obtained for the minimum successful execution sequence length in the non-epistemic traditional gossip literature: Farley and Proskurowski [23] proved that for a line topology network, the minimum number of pairwise communicative interactions for successful gossiping is $2n - 3$.

### 5.3.5 Protocol Descriptions on Star Topology Network

Listing 5.8 presents EGPL description of Protocol 2 on a star topology network. The network topology is described in lines 5-10 for a network of five agents $a, b, c, d, e$. In the network topology description, agent $a$ is at the center of the star network, and it is the only neighbour that any other agent has. In order to write the EGPL description that imposes the star topology network on other protocols we need to insert a network topology description similar to that in lines 5-10, for the agents in the scenario[¶], after describing the epistemic calling condition of the protocol.

```
1  begin
2       let ai call aj if {
3         ai knows (secret(ai) != secret(aj));
4       }
5       topology {
6         a neighbour b;
7         a neighbour c;
8         a neighbour d;
9         a neighbour e;
10      }
11 end
```

LISTING 5.8: **EGPL Description for Protocol 2 on Star Topology Network (Five Agents)**

### 5.3.6 Protocol Results on Star Topology Network

**Performance Analysis.** Protocol 3 features lower average execution sequence lengths and much lower standard extension sizes on a star topology network than Protocol 2,

[¶]A neighbour must be assigned to each agent in the scenario, where each agent is represented by a unique node of the network graph.

which indicates a greater adaptability of Protocol 3 over Protocol 2 with respect to a star topology network.

Protocol 4 and 5 remain non-terminating on a star topology network. Consider the following infinite execution sequence of Protocol 5 taken from a scenario with the network topology description shown in Listing 5.8: $ab; ac; ba; ca; ba; ca; \ldots$. That is, in the second round, agent $c$ learns the secret of agent $a$ and agent $b$ from agent $a$. Then, in the fourth round, agent $c$ again considers it possible that it will learn the secret of agent $d$ from agent $a$ (since agent $a$ may have called with agent $d$ in the third round), so agent $c$ calls agent $a$ in what proves to be a redundant fourth round call. In the fifth round, agent $b$ reasons as agent $c$ did in the fourth round, and likewise makes a redundant call to agent $a$ in the fifth round. And so does agent $c$ again in the sixth round, and the loop goes on infinitely in this sequence. Note that the given execution sequence is also an example of an infinite execution sequence for Protocol 4 (for example, if agent $c$ considers it possible that it will learn the secret of agent $d$ from a call with agent $a$, then it also considers it possible that it will learn some secret from the same call with agent $a$).

Looking at Table 5.11, Protocol 3 outperforms Protocol 2 in terms of scalability. And overall, for both protocols we see the scalability property gets better with increasing scenario size. (Note that Protocol 1 is not successful on a star topology network: the reason is analogous to that given in the preceding section, under the *Extension Analysis* for line topology network). Also refer to Appendix B for the empirical results for the scenario with six agents on a star topology network.

TABLE 5.9: **Protocol 2 on Star Topology Network.**

| Execution Sequence Length | Three Agents | Four Agents | Five Agents |
|---:|---:|---:|---:|
| 3 | 16 | | |
| 4 | | 0 | |
| 5 | | 288 | |
| 6 | | 768 | 0 |
| 7 | | | 12,288 |
| 8 | | | 61,440 |
| 9 | | | 196,608 |
| 10 | | | 294,912 |
| Standard Extension Size | 16 | 1,056 | 565,248 |
| Average Execution Sequence Length | 3 | 5.72727 | 9.36957 |
| Successful Sequences | 16 | 1,056 | 565,248 |
| % Successful Sequences | 100.00% | 100.00% | 100.00% |

TABLE 5.10: **Protocol 3 on Star Topology Network.**

| Execution Sequence Length | Three Agents | Four Agents | Five Agents |
|---:|---:|---:|---:|
| 3 | 16 | | |
| 4 | | 0 | |
| 5 | | 288 | |
| 6 | | 384 | 0 |
| 7 | | | 6,144 |
| 8 | | | 29,184 |
| 9 | | | 58,368 |
| 10 | | | 36,864 |
| Standard Extension Size | 16 | 672 | 130,560 |
| Average Execution Sequence Length | 3 | 5.57143 | 8.96471 |
| Successful Sequences | 16 | 672 | 130,560 |
| % Successful Sequences | 100.00% | 100.00% | 100.00% |

For the scalability property of the protocols on a star topology network, we summarise $\gamma$ for Protocol 2 and Protocol 3 in Table 5.11.

TABLE 5.11: **Protocol scalability on Star Topology Network.**

| Percentage Increase | Protocol 2 | Protocol 3 |
|---|---|---|
| $\gamma_{3,4}$ | 90.9090% | 85.7143% |
| $\gamma_{4,5}$ | 63.5957% | 60.9050% |
| $\gamma_{5,6}$ | 50.0584% | 47.6852% |

**Extension Analysis.** Similar to the line topology network for Protocol 1, there are no successful execution sequences on a star topology network for a scenario with more than two agents. And as on a line topology network this is easy to see: consider a gossip scenario with at least three agents $a, b, c, \ldots$ Assume the network topology shown in Listing 5.8 for these agents. So now agent $a$ is *in between* agent $b$ and agent $c$ in the network, such that, although the epistemic calling condition of Protocol 1 will be satisfied for agent $b$ to call agent $c$ (or for agent $c$ to call agent $b$), but since there is no direct link between agent $b$ and $c$, the only way for *both* agents to know each other's secrets is through the subsequence $ab; \ldots; ac; \ldots; ba$ or $ac; \ldots; ab; \ldots; ca$ (or their reverse call variants). But such subsequences do not comply with Protocol 1 (since no two agents agents can call each other more than once in an execution sequence of Protocol 1). So in Protocol 1 on a line topology network, one of agents $b$ and $c$ will never know all the secrets in the scenario. (See also Section 6.4).

For our experiments on the star topology network, both Protocols 4 and 5 feature the same number of successful execution sequences for $n = 3, 4, 5$.

TABLE 5.12: **Protocol 4 on Star Topology Network.**

| Execution Sequence Length | Three Agents | Four Agents | Five Agents |
|---|---|---|---|
| 3 | 16 | | |
| 4 | | 0 | |
| 5 | | 384 | |
| 6 | | 768 | 0 |
| 7 | | | 18,432 |
| 8 | | | 110,592 |
| 9 | | | 405,504 |
| 10 | | | 1,142,784 |
| Standard Extension Size | 16 | 1,188 | 1,718,388 |
| Average Execution Sequence Length | 3 | - | - |
| Successful Sequences | 16 | 1,152 | 1,677,312 |
| % Successful Sequences | 100.00% | 96.97% | 97.61% |

TABLE 5.13: **Protocol 5 on Star Topology Network.**

| Execution Sequence Length | Three Agents | Four Agents | Five Agents |
|---|---|---|---|
| 3 | 16 | | |
| 4 | | 0 | |
| 5 | | 384 | |
| 6 | | 768 | 0 |
| 7 | | | 18,432 |
| 8 | | | 110,592 |
| 9 | | | 405,504 |
| 10 | | | 1,142,784 |
| Standard Extension Size | 16 | 1,188 | 1,718,388 |
| Average Execution Sequence Length | 3 | - | - |
| Successful Sequences | 16 | 1,152 | 1,677,312 |
| % Successful Sequences | 100.00% | 96.97% | 97.61% |

On a star topology network, we compare some sample execution sequences. An example of a successful execution sequence in the standard extension of Protocol 2 but which is not in the standard extension of Protocol 3 is $ab; ac; ba; ad; ab; ac; ae; ab; ac; ad$.

The reason is that after the first two calls in the given sequence, in the third round, agent $b$ knows that agent $a$ must have made a call with some other agent and learnt some new secret (since agent $a$ is the only neighbour that each agent has, agent $a$ is involved in every call in this scenario). So although agent $b$ knows that it would learn some new secret by calling agent $a$ in the third round (de dicto), it does not know which new secret it would learn by calling $a$ in the third round (de re). The following is an example of an infinite execution sequence in the extension of Protocol 5: $ab; ac; ab; ca; ba; ca; ba; \ldots$. Such sequence shows how an agent at the centre of the star network (in this case agent $a$) can get into a loop of redundant calls with specific agents (in this case agent $b$ and $c$) because they always would consider it possible that they will learn some new secret from agent $a$, whereas agent $a$ keeps alternating between calling both of these agents.

**Comparison with Literature.** For Protocols 2-5, we do not obtain the shortest successful execution sequence length of $2n - 4$ for $n = 4, 5$. That is, for $n = 4$, there is no successful execution sequence of length 4; and for $n = 5$ there is no successful execution sequence of length 6. For the protocol in Listing 5.8 (that is, Protocol 2, for five agents, on a star topology network), the shortest successful execution sequence is of length seven (see Table 5.9). An example of such a sequence is $ac; ab; ad; ae; ad; ab; ac$. An example of a longest successful execution sequence for the same description is $ab; ad; ab; ae; ad; ab; ac; ad; ae; ab$. See also Table 5.10 for minimum successful execution sequence lengths for Protocol 3. These results for minimum successful execution sequence lengths all agree with the theoretical results obtained for the minimum successful execution sequence lengths in the non-epistemic traditional gossip literature: Harary and Schwenk [28] showed that if the underlying communication graph is a tree topology network then minimum length of a successful execution sequence is given by $2n-3$, for $n \geq 2$; Bumby [13] and Kleitman [36] confirmed this result by proving a stronger proposition - they showed that for any connected but incomplete (undirected) communication graph without a four-cycle, this minimum is $2n - 3$.

### 5.3.7 Protocol Descriptions on Binary Tree Topology Network

Listings 5.9 presents EGPL description of Protocol 2 on a binary tree topology network. The network topology is described in lines 5-10 for a network of five agents $a, b, c, d, e$. As before, in the epistemic calling condition (lines 2-4), `ai` and `aj` are agent name variables, in contrast to real agent names used to describe network topology. In the topology description, agent $a$ is at the root of the binary tree network, and it is the parent node of $b$ and $c$. Likewise $c$ is the parent node of $c$ and $e$, while $b$ has no child nodes. In order to write the EGPL description that imposes the binary tree topology on other protocols we would insert a network topology description similar to that in lines 5-10, after describing the epistemic calling condition of the protocol, where the network topology description is for the agents in the scenario.

```
1  begin
```

```
2        let ai call aj if {
3          ai knows (secret(ai) != secret(aj));
4        }
5        topology {
6          a neighbour b;
7          a neighbour c;
8          b neighbour d;
9          b neighbour e;
10       }
11   end
```

LISTING 5.9: **EGPL Description for Protocol 2 on Binary Tree Topology Network (Five Agents)**

### 5.3.8 Protocol Results on Binary Tree Topology Network

**Performance Analysis.** On a binary tree topology network, there are no successful execution sequences of Protocol 1 for a scenario with more than two agents. For the scenario with four agents on the binary tree topology network, Protocol 2 and Protocol 3 have the same standard extension sizes and average execution sequence lengths. Notice that the average execution sequence length of Protocol 2 for the scenario with four agents is the same for both the star topology network and the binary tree topology network (see Table 5.9 and Table 5.14).

TABLE 5.14: **Protocol 2 on Binary Tree Topology Network.**

| Execution Sequence Length | Three Agents | Four Agents | Five Agents |
|---:|---:|---:|---:|
| 3 | 16 | | |
| 4 | | 0 | |
| 5 | | 192 | |
| 6 | | 512 | 0 |
| 7 | | | 4,992 |
| 8 | | | 30,080 |
| 9 | | | 75,904 |
| 10 | | | 100,864 |
| Standard Extension Size | 16 | 704 | 211,840 |
| Average Execution Sequence Length | 3 | 5.72727 | 9.28701 |
| Successful Sequences | 16 | 704 | 211,840 |
| % Successful Sequences | 100.00% | 100.00% | 100.00% |

From Table 5.14 and Table 5.15 we see that for a scenario with five agents, Protocol 3 exhibits a lower average execution sequence length than Protocol 2 on a binary tree topology network. From our experiments, Protocol 2 features lower average execution sequence lengths and lower standard extension sizes on the binary tree topology network than it does on the star topology network. On the contrary, Protocol 3 features lower average execution sequence lengths on the star topology network than it does on the binary tree topology network. For a scenario with five agents, Protocol 3 features a larger standard extension size on the star topology network than it does on the binary tree topology network, but the reverse is the case for a scenario with four agents. Therefore from our experiments, Protocol 2 is more adaptable on the binary tree topology network than on the star topology network with respect to both time efficiency and space

efficiency, and Protocol 3 is more adaptable on the star topology network than on the binary tree topology network with respect to time efficiency.

TABLE 5.15: **Protocol 3 on Binary Tree Topology Network.**

| Execution Sequence Length | Three Agents | Four Agents | Five Agents |
|---|---|---|---|
| 3 | 16 | | |
| 4 | | 0 | |
| 5 | | 192 | |
| 6 | | 512 | 0 |
| 7 | | | 4,992 |
| 8 | | | 20,480 |
| 9 | | | 51,456 |
| 10 | | | 48,896 |
| Standard Extension Size | 16 | 704 | 125,824 |
| Average Execution Sequence Length | 3 | 5.72727 | 9.14649 |
| Successful Sequences | 16 | 704 | 125,824 |
| % Successful Sequences | 100.00% | 100.00% | 100.00% |

For the scalability property of the protocols on a binary tree topology network, we summarise $\gamma$ for Protocol 2 and Protocol 3 in Table 5.16.

TABLE 5.16: **Protocol scalability on Binary Tree Topology Network.**

| Percentage Increase | Protocol 2 | Protocol 3 |
|---|---|---|
| $\gamma_{3,4}$ | 90.9090% | 90.9090% |
| $\gamma_{4,5}$ | 62.1542% | 59.7007% |
| $\gamma_{5,6}$ | 48.8971% | 50.4550% |

Looking at Table 5.16, Protocol 3 outperforms Protocol 2 in terms of scalability. And overall, for both protocols we see the scalability property gets better with increasing scenario size. (Again, note that Protocol 1 is not successful on a star topology network). So, similar to the scalability results on complete, line and star topology networks, Protocol 3 proves to be more scalable on a binary tree topology network than Protocol 2. Also refer to Appendix B for the empirical results for the scenario with six agents on the binary tree topology network.

TABLE 5.17: **Protocol 4 on Binary Tree Topology Network.**

| Execution Sequence Length | Three Agents | Four Agents | Five Agents |
|---|---|---|---|
| 3 | 16 | | |
| 4 | | 0 | |
| 5 | | 192 | |
| 6 | | 768 | 0 |
| 7 | | | 7,424 |
| 8 | | | 67,072 |
| 9 | | | 278,016 |
| 10 | | | 807,424 |
| Standard Extension Size | 16 | 974 | 1,194,434 |
| Average Execution Sequence Length | 3 | - | - |
| Successful Sequences | 16 | 960 | 1,159,936 |
| % Successful Sequences | 100.00% | 98.56% | 97.11% |

Protocol 4 and 5 remain non-terminating on a binary tree topology network, with both protocols featuring the same number of successful and unsuccessful execution sequences, as well as the same standard extension size. An example of a non-terminating sequence in Protocol 4 on the binary tree topology network shown in Listing 5.9 is $bd; be; bd; eb; db; eb; db; \ldots$, with the subsequence $eb; db$ repeating infinitely.

**Extension Analysis.** As on the star topology network, we do not obtain the shortest successful execution sequence length for $n = 4, 5$, for Protocols 2-5 on the binary tree topology network. The shortest successful execution sequence for a scenario with four agents is of length five (rather than $2n - 4 = 4$), and the shortest successful execution sequence for a five-agent scenario is of length seven. An example of a shortest successful execution sequence for the description given in Listing 5.9 is $ac; ab; bd; be; ab; ac; bd$. An example of a longest successful execution sequence (length of $n(n-2)/2$) is: $ab;db;ca;ba;bd;ac;be;ab;db;ca$. For the description given in Listing 5.9 the following execution sequence is in Protocol 2 but not in Protocol 3: $be; ca; db; eb; ab; ac; bd; be$. After the second and third rounds, in third round, agent $e$ knows that given the network topology, agent $b$ must have made a call (either with agent $d$ or with agent $a$) in the second round or in the third round - if agent $b$ did not call with agent $a$ in the second round it was because agent $b$ was calling agent $d$, or because agent $a$ was calling with agent $c$ in the second round in which case agent $b$ will call agent $a$ or agent $d$ in the third round. Moreover in the fourth round agent $e$ does not know a particular secret that it would learn by calling agent $b$. Finally, the average execution sequence length of Protocol 2 is better on the binary tree topology network than on the star topology network.

**Comparison with Literature.** Similar to how the results for the star topology network, results shown in Tables 5.14 and 5.15 for minimum successful execution sequence lengths also agree with the theoretical results obtained for the minimum successful execution sequence lengths in the non-epistemic traditional gossip literature.

### 5.3.9 Protocol Descriptions on Circle Topology Network

Listings 5.10 presents EGPL description of Protocols 2 on a circle topology network. The network topology is described in lines 5-10 for a network of five agents $a, b, c, d, e$ and the epistemic calling condition is given in lines 2-4, where `ai` and `aj` are agent name variables, in contrast to real agent names used to describe the network topology. In the network topology description, all the agents each have exactly two neighbours, and any two agents have exactly one neighbour in common. In order to write the EGPL description that imposes the circle topology network on other protocols we would insert a network topology description similar to that in lines 5-10, after describing the epistemic calling condition of the protocol, where the network topology description is for the agents in the scenario.

```
1  begin
2       let ai call aj if {
3         ai knows (secret(ai) != secret(aj));
4       }
5       topology {
6         a neighbour b;
7         b neighbour c;
```

```
 8           c neighbour d;
 9           d neighbour e;
10           e neighbour a;
11         }
12   end
```

LISTING 5.10: **EGPL Description for Protocol 2 on Circle Topology Network (Five Agents)**

### 5.3.10 Protocol Results on Circle Topology Network

We present the results of our experiments with epistemic gossip protocols in circle topology networks.

**Extension Analysis.** Protocol 1 is successful only for $n = 3$. For $n = 4$, Protocol 1 is no longer successful. It then has 128 successful and 16 unsuccessful execution sequences. For $n \geq 4$, Protocol 1 is not successful on the circle topology network (we prove this claim later in Chapter 6, see Proposition 6.55). Between Protocols 2 and 3, we observe that Protocol 3 offers the least standard extension size on a circle topology network, although for Protocol 2 we obtain $1,115,200$ successful execution sequences out of a standard extension size of $1,115,240$, for the scenario with five agents, whereas Protocol 3 for the five agent scenario yields $827,840$ successful sequences out of a standard extension consisting of $827,940$ execution sequences.

TABLE 5.18: **Protocol 1 on Circle Topology Network.**

| Execution Sequence Length | Three Agents | Four Agents |
|---:|---:|---:|
| 3 | 24 | |
| 4 | | 128 |
| 5 | | 0 |
| 6 | | 0 |
| Standard Extension Size | 24 | 144 |
| Average Execution Sequence Length | 3 | - |
| Successful Sequences | 24 | 128 |
| % Successful Sequences | 100.00% | 88.89% |

TABLE 5.19: **Protocol 2 on Circle Topology Network.**

| Execution Sequence Length | Three Agents | Four Agents | Five Agents |
|---:|---:|---:|---:|
| 3 | 96 | | |
| 4 | | 128 | |
| 5 | | 1,920 | |
| 6 | | 5,248 | 0 |
| 7 | | | 53,440 |
| 8 | | | 248,320 |
| 9 | | | 432,480 |
| 10 | | | 380,960 |
| Standard Extension Size | 96 | 7,296 | 1,115,240 |
| Average Execution Sequence Length | 3 | 5.70175 | - |
| Successful Sequences | 96 | 7,296 | 1,115,200 |
| % Successful Sequences | 100.00% | 100.00% | 99.9964% |

Both Protocol 4 and 5 offer similar properties on a circle topology network - Table 5.21 for Protocol 4 is identical to that for Protocol 5. Protocol 2 (whose standard extension is greater than that of Protocol 3) has about 68 times larger standard extension than Protocols 4 and 5.

TABLE 5.20: **Protocol 3 on Circle Topology Network.**

| Execution Sequence Length | Three Agents | Four Agents | Five Agents |
|---|---|---|---|
| 3 | 96 | | |
| 4 | | 128 | |
| 5 | | 1,920 | |
| 6 | | 5,248 | 0 |
| 7 | | | 39,360 |
| 8 | | | 188,960 |
| 9 | | | 319,040 |
| 10 | | | 280,480 |
| Standard Extension Size | 96 | 7,296 | 827,940 |
| Average Execution Sequence Length | 3 | 5.70175 | - |
| Successful Sequences | 96 | 7,296 | 827,840 |
| % Successful Sequences | 100.00% | 100.00% | 99.99% |

TABLE 5.21: **Protocol 4 on Circle Topology Network.**

| Execution Sequence Length | Three Agents | Four Agents | Five Agents |
|---|---|---|---|
| 3 | 96 | | |
| 4 | | 128 | |
| 5 | | 2,816 | |
| 6 | | 13,312 | 0 |
| 7 | | | 133,120 |
| 8 | | | 1,824,000 |
| 9 | | | 12,474,880 |
| 10 | | | 61,048,320 |
| Standard Extension Size | 96 | 16,524 | 76,100,860 |
| Average Execution Sequence Length | 3 | - | - |
| Successful Sequences | 96 | 16,256 | 75,480,320 |
| % Successful Sequences | 100.00% | 98.38% | 99.18% |

For the network topology described in Listing 5.10 we provide the following example sequences. An example of a successful execution sequence in the standard extension of Protocol 2 but which is not in the standard extension of Protocol 3 is $ab; ed; dc; ea; ed; cb; ab$ (in the last round agent $a$ knows that it would learn some secret in the call with agent $b$, but it is unsure which secret it would learn from agent $b$). An example of a successful execution sequence in the standard extension of Protocol 3 but which is not in the standard extension of Protocol 2 is $ab; ae; ab; cd; de; bc; ea$. An example of an unsuccessful execution sequence in the standard extension of Protocol 2 is $ab; ae; bc; ed; cd; ea; bc; ab$. An example of an unsuccessful execution sequence in the standard extension of Protocol 3 is $ab; ae; ab; bc; ba; cd; de$. An example of a successful execution sequence in the standard extension of Protocol 5 is $ab; cd; ae; bc; ab; bc; de$. An example of an unsuccessful execution sequence in the standard extension of Protocol 5 is $ab; ae; ab; ae; ab; ae; ab; ae; \ldots$ We refer the reader also to Section 6.5 where we provide a detailed demonstration of an unsuccessful execution sequence for Protocol 2 and Protocol 3.

## 5.4 A Discussion About Equivalence Notions

In order to generate the extension of an epistemic gossip protocol, the EGP tool interprets the epistemic calling condition of a given protocol on the nodes of an epistemic tree, beginning from the node representing the initial gossip situation (see Section 4.2.4). The equivalence notion that we have adopted in this thesis for the interpretation of epistemic formulas on an epistemic tree is given in Definition 4.7. Informally, that equivalence notion says that an agent considers two execution sequences to be equivalent if it calls the same other agent in each corresponding call of both execution sequences (or, if it was

not involved in both corresponding calls), and if it knows the same secrets at the end of each corresponding call in both sequences.

However, in Section 4.5 (Chapter 4), we defined an alternative equivalence notion that could be used in the interpretation of epistemic formulas on an epistemic tree (see Definition 4.40). In the alternative equivalence notion, an agent considers two execution sequences to be equivalent if it calls the same other agent in each corresponding call of both execution sequences (or, if it was not involved in both corresponding calls), learns the same secrets from that other agent in the corresponding calls of both sequences, and knows the same secrets at the end of each corresponding call in both execution sequences.

Obviously, the equivalence notion used by the EGP tool can alter the extension of an epistemic gossip protocol. Let the equivalence notion given in Definition 4.7 be called *Equivalence Notion 1*, and let the equivalence notion given in Definition 4.40 be called *Equivalence Notion 2* or *Alternative Equivalence Notion*. Before we close this chapter, we want to justify our choice of Equivalence Notion 1 as the running notion of equivalence in this thesis. We do this by briefly discussing some preliminary empirical results obtained by using Equivalence Notion 2, and comparing them with some of the results obtained by using Equivalence Notion 1 in the EGP tool (note that all the results that we have discussed so far are based on Equivalence Notion 1).

Tables 5.22 and 5.23 show the results obtained by using Equivalence Notion 2, for Known Information Growth de Dicto (Protocol 2) and Known Information Growth de Re (Protocol 3), respectively. We now compare results in Table 5.22 to the results in Table 5.2 due to Equivalence Notion 1. We also compare results in Table 5.23 to the results in Table 5.3 due to Equivalence Notion 1.

TABLE 5.22: **Protocol 2 on Complete Topology Network.**

| Execution Sequence Length | Three Agents | Four Agents | Five Agents |
|---:|---:|---:|---:|
| 3 | 96 | | |
| 4 | | 384 | |
| 5 | | 16,896 | |
| 6 | | 83,712 | 195,840 |
| 7 | | | 8,664,960 |
| 8 | | | 87,943,680 |
| 9 | | | 375,682,560 |
| 10 | | | 927,025,920 |
| Standard Extension Size | 96 | 100,992 | 1,399,512,960 |
| Average Execution Sequence Length | 3 | 5.8251 | 9.58675 |
| Successful Sequences | 96 | 100,992 | 1,399,512,960 |
| % Successful Sequences | 100.00% | 100.00% | 100.00% |

TABLE 5.23: **Protocol 3 on Complete Topology Network.**

| Execution Sequence Length | Three Agents | Four Agents | Five Agents |
|---:|---:|---:|---:|
| 3 | 96 | | |
| 4 | | 384 | |
| 5 | | 15,744 | |
| 6 | | 80,256 | 149,760 |
| 7 | | | 6,871,680 |
| 8 | | | 62,991,360 |
| 9 | | | 331,944,960 |
| 10 | | | 769,326,720 |
| Standard Extension Size | 96 | 96,384 | 1,171,284,480 |
| Average Execution Sequence Length | 3 | 5.82869 | 9.59093 |
| Successful Sequences | 96 | 96,384 | 1,171,284,480 |
| % Successful Sequences | 100.00% | 100.00% | 100.00% |

For both equivalence notions, the results remain the same for a scenario with three agents. For a scenario with four agents and five agents, the sizes of the standard extension of Known Information Growth de Dicto grew, respectively, by 24.6% and 83.5% when we switched from Equivalence Notion 1 to Equivalence Notion 2. This indicates an exponential growth in standard extension size for switching from Equivalence Notion 1 to Equivalence Notion 2 for Known Information Growth de Dicto, on the complete topology network.

For Known Information Growth de Re, we witness the same exponential growth in switching from Equivalence Notion 1 to Equivalence Notion 2. We obtain a 41.4% increase in the standard extension size for four agents, and a 116% increase in the standard extension size for five agents, due to the switch from Equivalence Notion 1 to Equivalence Notion 2.

Regarding average execution sequence lengths, observe that in the switch from Equivalence Notion 1 to Equivalence Notion 2, the increase in the standard extension sizes are mainly due to an increase in the number of longer execution sequences. So, naturally we also obtain an increase in the average execution sequence lengths by switching from Equivalence Notion 1 to Equivalence Notion 2.

More empirical results from Equivalence Notion 2 can be found in Appendix C of the thesis. Note that we have not included results for Learn New Secrets (Protocol 1), since they are the same for both equivalence notions (the reason being that the epistemic calling condition for Learn New Secrets can be checked by looking at only the set of secrets known (or not known) by an agent in the current gossip situation, as such, the variation between Equivalence Notion 1 and Equivalence Notion 2 does not affect the calls that are possible at each node of the gossip tree, and consequently does not affect the extension of the Learn New Secrets protocol).

We have also not included results for Possible Information Growth de Dicto (Protocol 4) and Possible Information Growth de Re (Protocol 5) for Equivalence Notion 2, since they remain non-terminating irrespective of whether the equivalence notion used is Equivalence Notion 1 or Equivalence Notion 2. An example which shows that Possible Information Growth de Dicto and Possible Information Growth de Re are non-terminating, is as follows. Consider a scenario with four agents $a, b, c, d$. For the first call, let agent $a$ call agent $b$; for the second call let agent $c$ call agent $d$; then for the third call, let agent agent $a$ call agent $b$ again since agent $a$ considers it possible that it will learn agent $c$'s secret from agent $b$ (agent $a$ considers it possible that the second call of the execution sequence was in fact between agent $b$ and agent $c$), but now we see that both agent $a$ and agent $b$ learn no new secret by calling each other again; for the fourth call, let agent $c$ call agent $d$ again, since agent $c$ considers it possible that it will learn agent $b$'s secret from agent $d$ (agent $c$ considers it possible that the third call was in fact between agent $b$ and agent $d$), but again we see that both agents learn no new secret from the call; for the fifth and sixth call, agent $a$ and agent $c$ respectively reason as they did for the third and fourth calls; and this goes on indefinitely for subsequent calls.

Notice from the tables shown in Appendix C (and Appendix B) that on a line topology network, both equivalence notions yield the same standard extension sizes and average execution sequence lengths for Known Information Growth de Dicto. The same is true also for Known Information Growth de Re. For a circle topology network, whereas there is a general increase in the standard extension sizes when we switch from Equivalence Notion 1 to Equivalence Notion 2, both protocols remain unsuccessful for more than four agents, under both Equivalence Notion 1 and Equivalence Notion 2. On the star and binary tree topology networks, Known Information Growth de Dicto maintain the same standard extension sizes and average execution sequence lengths across both equivalence notions. The same is also true for Known Information Growth de Re on the star and binary tree topology networks.

## 5.5 Conclusion

In this chapter we analysed epistemic gossip protocols using the EGP tool. We presented empirical results obtained for Learn New Secrets (Protocol 1), Known Information Growth de Dicto (Protocol 2), Known Information Growth de Re (Protocol 3), Possible Information Growth de Dicto (Protocol 4) and Possible Information Growth de Re (Protocol 5). The performance of the gossip protocols were studied on the complete, line, star, binary tree and circle topology networks. We compared the performance of these protocols in terms of their time and space efficiency (given by the average execution sequence length and standard extension size of these protocols). We also compared the protocols with respect to their scalability and adaptability properties.

Our experiments show that the complete topology network is the setting where the most of our protocols are successful. Both Known Information Growth de Dicto and Known Information Growth de Re are seen to be successful on all the investigated network topologies except the circle topology network where they are successful only for the empirical cases of $n = 3$ and $n = 4$ agents. On the other hand, both Possible Information Growth de Dicto and Possible Information Growth de Re are successful only for the empirical case of $n = 3$ agents, but on all the investigated network topologies. Also, Learn New Secrets is successful only on the complete topology network, out of all the investigated network topologies.

The line topology network gives rise to the minimum standard extension size for all the investigated protocols. The reduction in standard extension size on a line topology network is most pronounced when compared to the standard extension size of the same protocol on a complete topology network, since, from our experiments, the complete topology network gives rise to the greatest standard extension sizes out of all the investigated network topologies. For example, for Known Information Growth de Dicto, the standard extension size reduced by a factor of 6498 for a scenario with five agents, when the setting was moved from a complete topology network to a line topology network. For the Known Information Growth protocols, the binary tree topology network gives rise

to the next higher standard extension size after the line topology network; the standard extension size obtained on the binary tree topology network is then followed by that obtained on the star topology network, which is in turn followed by the standard extension size obtained on the circle topology network, in order of increasing standard extension sizes, for the investigated cases of number of agents $n$.

In terms of average execution sequence length, Learn New Secrets gives rise to the least average execution sequence length out of all the investigated protocols, although it is successful only on a complete topology network. Although the circle topology network gives rise to the least average execution sequence length for the Known Information Growth protocols, both Known Information Growth de Dicto and Known Information Growth de Re are successful only for the empirical cases of $3 \leq n \leq 4$ agents on a circle topology network (both Known Information Growth de Dicto and Known Information Growth de Re have the same average execution sequence length for $3 \leq n \leq 4$ agents on a circle topology network).

For the network topologies on which Known Information Growth de Dicto and Known Information Growth de Re are successful, namely, the complete, star and binary tree topology networks, we observe that when we run Known Information Growth de Re on a star topology network, we obtain the lowest average execution sequence lengths, out of both Known Information Growth protocols. On the other hand we observe that when we run Known Information Growth de Dicto on a complete topology network, we obtain the highest average execution sequence length out of all the investigated protocols and network topologies. Our empirical results also indicate that on a line topology network, the standard extension of Known Information Growth de Dicto is equivalent to that of Known Information Growth de Re (we treat this formally in Section 6.3; see Corollary 6.39).

Of all the investigated protocols, Known Information Growth de Dicto and Known Information Growth de Re are successful across the most number of network topologies. They are successful on the complete, line, star, and binary tree topology networks, for the empirical cases of $3 \leq n \leq 6$ agents (for the complete topology network, the empirical cases are for $3 \leq n \leq 5$ agents). The Known Information Growth protocols are also successful on the circle topology network but for the empirical cases of $3 \leq n \leq 4$ agents. No other investigated protocol exhibits such adaptation across various network topologies.

Regarding a future research work relating to this chapter, an interesting investigation would be to use the EGP tool to try to discover other network topologies that significantly improve the performance of our epistemic gossip protocols or other epistemic gossip protocols. Other network topologies that could be studied are various kinds of grid networks, and directed network graphs.

# Chapter 6

# Epistemic Gossip Protocols and Network Topologies

## 6.1 Introduction

In gossip scenarios, as typical in the study of multiagent systems [70], we assume that the agents are connected via a communication network. In chapter three we described a number of epistemic gossip protocols with the implicit assumption that the underlying network of agents is a complete network graph. In Chapter 5 we called such a network a complete topology network.

However, complete topology networks are often expensive to build and maintain. Moreover, many kinds of existing networks are not complete, and need not be complete. In this chapter we loosen the constraint of executing the epistemic gossip protocols on only complete topology networks. We explore the properties of our gossip protocols on some other connected, though incomplete, network topologies. Specifically we consider complete topology network (Section 6.2), tree topology networks (Sections 6.3 and 6.4) and circle topology network (Section 6.5). For each of the network topologies, we present proofs for some of the properties of Learn New Secrets, Known Information Growth de Dicto, Known Information Growth de Re, Possible Information Growth de Dicto and Possible Information Growth de Re.

### Ancillary Assumptions and Definitions

In Chapter 3, for an epistemic gossip protocol $\Pi$, an agent $a_i$ calls another agent $a_j$ only if the epistemic calling condition for call $a_ia_j$ given by $\Pi$ holds in the gossip situation. That is, there was an implicit assumption that there is a direct communication link (or graph edge) between any pair of agents $a_i$ and $a_j$ in the underlying communication graph. In this chapter however, we consider various network topologies of the underlying communication graph. Therefore, the condition for any pair of agents to call each other is then as follows: an agent $a_i$ calls another agent $a_j$ only if the epistemic calling condition for call $a_ia_j$ given by $\Pi$ holds in the gossip situation *and* agent $a_i$ and $a_j$ are neighbours in

the underlying network graph. We assume that all the network graphs are bidirectional. And we assume that the network topology is common knowledge among all the agents in the gossip scenario. We also assume that the protocol for each agent is common knowledge among all the agents in the gossip scenario. Similar to Chapters 4 and 5, we assume throughout this chapter that the mode of call is the private synchronous mode (that is, the 0-mode). We also make use of the following definitions.

**Definition 6.1** (Redundant Call)**.** A call $a_i a_j$ is redundant if neither $a_i$ nor $a_j$ will learn any new secret from the $a_i a_j$ call.

**Definition 6.2** (Regular Execution Sequence)**.** A regular execution sequence of calls is that in which there are no redundant calls.

**Definition 6.3** (Regular Protocol)**.** An epistemic gossip protocol is regular if the epistemic calling condition of the protocol is such that no pair of agents make a redundant call in any execution sequence of the protocol.

**Definition 6.4** (Calling Condition)**.** Given a network of agents in a gossip scenario, and given an epistemic gossip protocol $\Pi$, then at any gossip situation or any history, the *calling condition* for an agent $a_i$ to call another agent $a_j$ is two-fold, as follows:

1. The *epistemic calling condition* for agent $a_i$ to call agent $a_j$ under protocol $\Pi$ must be satisfied (see Definition 3.23), and

2. Agent $a_i$ and agent $a_j$ must be neighbours on the network graph.

We will also make use of the definition of *successful execution sequence*, *successful protocol*, *terminating execution sequence* and *terminating protocol*, and other definitions as given in Chapter 4, except where we explicitly mention otherwise. For instance, to account for the underlying network topology, we modify the definition of an execution sequence (or history) of an epistemic gossip protocol and that of a terminal execution sequence, as follows.

**Definition 6.5** (Execution Sequence of an Epistemic Gossip Protocol)**.** Given an epistemic gossip protocol $\Pi$ and given any history $h$. Let $\varphi_\Pi(a_i, a_j)$ be the epistemic calling condition for some agent $a_i$ to call another agent $a_j$ under protocol $\Pi$. Then $h$ is an execution sequence of $\Pi$ if and only if one of the following conditions hold:

- $h = e$

- $h = h'; a_i a_j$, and $h' \in H^{h'}$, and $\mathcal{C}(h'), h' \models \varphi_\Pi(a_i, a_j)$ and [$a_i$ and $a_j$ are neighbours on the network graph], where the epistemic context $\mathcal{C}(h')$ of $h'$, and $H^{h'}$ are as defined in Definition 4.12.

Note that by Definition 6.5, the extension of an epistemic gossip protocol is determined also by the given network topology. As a result, following Definition 4.16 for a

gossip tree, the set of histories for the gossip tree of an epistemic gossip protocol is determined also by the given network topology. So we can say that by following Definition 6.5, a gossip tree *respects* the given network topology of the network of agents. In this chapter, whenever we refer to a gossip tree we mean that which respects the underlying network topology.

**Definition 6.6** (Terminal Execution Sequence)**.** Given an epistemic gossip protocol $\Pi$, let $\mathcal{T}_g$ be the gossip tree for $\Pi$. Let $\varphi_\Pi(a_i, a_j)$ be the epistemic calling condition for $a_i$ to call $a_j$, where $a_i, a_j \in \mathsf{Ag}$. Then, an execution sequence $\sigma \in \mathcal{T}_g$ is terminal if and only if for every pair $a_i, a_j \in \mathsf{Ag}$, it is the case that: either $\mathcal{T}_g, \sigma \models \neg\varphi_\Pi(a_i, a_j)$, or $a_i$ and $a_j$ are *not* neighbours on the network graph.

**Definition 6.7** (Fresh Call and Fresh Caller)**.** Given any call sequence $\sigma$, and any call $a_i a_j$ in $\sigma$, then $a_i a_j$ contains a *fresh caller* if such $a_i a_j$ call is the first call that either $a_i$ or $a_j$ is involved in, in the execution sequence $\sigma$. A call is a fresh call if it contains at least one fresh caller.

### 6.1.1 Properties of Regular Protocols

We now consider some properties of regular protocols and regular execution sequences. The following observation is convenient.

*Observation* 6.8. Let the epistemic calling condition for agent $a_i$ to call $a_j$, for Learn New Secrets, Known Information Growth de Dicto and Known Information Growth de Re be $\varphi_{\Pi_1}(a_i, a_j)$, $\varphi_{\Pi_2}(a_i, a_j)$ and $\varphi_{\Pi_3}(a_i, a_j)$, respectively. Recall those conditions are as follows:

$$\varphi_{\Pi_1}(a_i, a_j) = K_{a_i} \neg Kw_{a_i} A_j \tag{6.1}$$

$$\varphi_{\Pi_2}(a_i, a_j) = K_{a_i} \bigvee_{a_k \in \mathsf{Ag}} (Kw_{a_i} A_k \nabla Kw_{a_j} A_k) \tag{6.2}$$

$$\varphi_{\Pi_3}(a_i, a_j) = K_{a_i} \bigvee_{a_k \in \mathsf{Ag}} K_{a_i}(Kw_{a_i} A_k \nabla Kw_{a_j} A_k) \tag{6.3}$$

Given any gossip tree $\mathcal{T}_g$, from Observation 4.26 we see that:

$$\mathcal{T}_g \models \varphi_{\Pi_1}(a_i, a_j) \rightarrow \varphi_{\Pi_3}(a_i, a_j) \quad \text{and} \quad \mathcal{T}_g \models \varphi_{\Pi_3}(a_i, a_j) \rightarrow \varphi_{\Pi_2}(a_i, a_j) \tag{6.4}$$

Let us define the condition $\varphi(a_i, a_j)$ as:

$$\varphi(a_i, a_j) = \bigvee_{a_k \in \mathsf{Ag}} (Kw_{a_i} A_k \nabla Kw_{a_j} A_k) \tag{6.5}$$

Note that for each of the conditions $\varphi_{\Pi_1}(a_i, a_j)$, $\varphi_{\Pi_2}(a_i, a_j)$ and $\varphi_{\Pi_3}(a_i, a_j)$, we have that:

$$\mathcal{T}_g \models \varphi_{\Pi_1}(a_i, a_j) \rightarrow \varphi(a_i, a_j) \text{ and } \mathcal{T}_g \models \varphi_{\Pi_2}(a_i, a_j) \rightarrow \varphi(a_i, a_j) \text{ and } \mathcal{T}_g \models \varphi_{\Pi_3}(a_i, a_j) \rightarrow \varphi(a_i, a_j) \quad \text{(by veridicality)} \quad (6.6)$$

In other words: if the calling condition for agent $a_i$ to call $a_j$ is satisfied for any of the three protocols, then there is some secret $A_k$ that is known by only one of those two agents. Furthermore,

$$\mathcal{T}_g \models \neg Kw_{a_i} A_j \rightarrow \varphi_{\Pi_1}(a_i, a_j) \quad \text{(by negative Introspection)} \quad (6.7)$$

Hence, from (6.4):

$$\mathcal{T}_g \models \neg Kw_{a_i} A_j \rightarrow \varphi_{\Pi_2}(a_i, a_j) \text{ and } \mathcal{T}_g \models \neg Kw_{a_i} A_j \rightarrow \varphi_{\Pi_3}(a_i, a_j) \quad \text{(by propositional logic)} \quad (6.8)$$

And, from (6.8) and (6.6):

$$\mathcal{T}_g \models \neg Kw_{a_i} A_j \rightarrow \varphi(a_i, a_j) \quad \text{(by propositional logic)} \quad (6.9)$$

**Proposition 6.9.** *Given any network of agents, Known Information Growth de Re, Known Information Growth de Dicto and Learn New Secrets are regular protocols.*

*Proof.* Consider any network of agents, and take any execution sequence $\sigma$ of any of the three protocols. Let the gossip tree for the protocol be $\mathcal{T}_g$, and let $\sigma = \ldots; a_i a_j; \ldots$, where $a_i a_j$ is an arbitrary call in $\sigma$. Let $\sigma'$ be the longest history of calls preceding the $a_i a_j$ call in $\sigma$. Then the calling condition for agent $a_i$ to call $a_j$ in the $a_i a_j$ call must have been true at $\sigma'$. From Definition 6.4, it follows that $a_i$ and $a_j$ are neighbours on the network graph, and the epistemic calling condition for agent $a_i$ to call agent $a_j$ holds at $(\mathcal{T}_g, \sigma')$. Hence from Observation 6.8 (expression (6.6)) it also follows that there is some secret $A_k$ known by exactly one of agents $a_i$ and $a_j$ at $(\mathcal{T}_g, \sigma')$, such that the other agent learnt this secret $A_k$ in the $a_i a_j$ call, and hence this call is not redundant. Since the call $a_i a_j$ is chosen arbitrarily from $\sigma$, and since $\sigma$ is an arbitrary execution sequence of any of the three given protocols, then none of the execution sequences of the three protocols contain redundant calls, and therefore these three protocols are regular. □

**Lemma 6.10.** *Given an epistemic gossip scenario consisting of $n$ agents, and given any network topology, then in any successful execution sequence, there are at most $n-1$ fresh calls.*

*Proof.* Consider any network of agents, and consider any successful execution sequence $\sigma$. Suppose, towards a contradiction, that there are more than $n-1$ fresh calls in $\sigma$, say, without loss of generality, that there are $n$ fresh calls in $\sigma$. Since each of the fresh

calls must contain at least one fresh caller, and the first call in $\sigma$ must contain exactly two fresh callers (since it is the first time that pair of agents make any call), it follows that there must be at least $n+1$ agents in the scenario, which then contradicts our assumption that there are $n$ agents in the scenario.

Finally to show that $n-1$ fresh calls are possible in $\sigma$, consider a case, given below, where a designated agent $a_1$ calls each of all the other agents in turn at the beginning of $\sigma$:

$$\sigma = a_1a_2; \ldots; a_1a_n; \ldots$$

It is clear in the given example $\sigma$, that the first $n-1$ calls are the fresh calls. □

**Lemma 6.11.** *For a gossip scenario, given any regular and successful execution sequence $\sigma$, and any agent $a_i \in \mathsf{Ag}$, then exactly $n-1$ calls are required in $\sigma$ to spread the secret of agent $a_i$ to all the other agents.*

*Proof.* Prior to the first call that $a_i$ makes in $\sigma$, there are $n-1$ agents that do not know the secret of agent $a_i$. Any call in $\sigma$ in which the secret of $a_i$ is learnt for the first time must be between an agent who knows the secret of $a_i$ and another agent who does not know the secret of agent $a_i$. Thus for the $n-1$ agents who did not initially know the secret of $a_i$, we need $n-1$ calls to inform them all of the secret of $a_i$. □

**Proposition 6.12.** *In a gossip scenario of $n$ agents, the maximum length of any regular and successful execution sequence is $n(n-1)/2$.*

*Proof.* Let $f : \mathbb{N} \to \mathbb{N}$ be a function that returns the maximum length of any regular and successful execution sequence of calls given the number $n$ of agents in the gossip scenario. Then:

$$f(1) = 0; \qquad \text{(no calls are made)}$$

$$f(2) = 1; \qquad \text{(only one call is made)}$$

For $n = 3$, any two agents have to call each other in the first call; in the second call, for some agent to learn some new secret, one of the agents who called in the first call has to call with the other agent who did not take part in the first call; in the third call, for some agent to learn some new secret, one of the callers in the second call have to call with the agent who was not in the second call. After this third call, all three agents know all the secrets in the scenario, and so no further calls are required. So exactly three calls are made in any regular and successful execution sequence. Therefore for $n = 3$:

$$f(3) = 3;$$

Now suppose that $f(n-1)$ is the maximum length of a regular and successful execution sequence in a gossip scenario with $n-1$ agents, and there is an execution sequence $\sigma$ consisting of $f(n-1)$ calls after which all the $n-1$ agents know each other's secrets. Suppose that we introduce a new agent $a_n$ into such a scenario after executing $\sigma$, then

by Lemma 6.11, exactly $n-1$ additional calls are required to spread the secret of $a_n$ to all the other agents in the scenario, given that some agent must learn some new secret in each call. Note that the first call that follows $\sigma$ must be a call involving $a_n$, otherwise the call will be redundant. Note also that in that first call after $\sigma$, agent $a_n$ learns the secret of every other agent. Furthermore, from Lemma 6.11, for each of the initial $n-1$ agents, $n-1$ calls are required for all the other agents to know their secret. But after $\sigma$, those initial $n-1$ agents know all of each other's secrets, so again, from Lemma 6.11, for each agent $a'$ in the initial $n-1$ agents, $\sigma$ must contain exactly $n-2$ calls in which an agent learnt the secret of $a'$ for the first time. So, at the first call following $\sigma$ in which $a_n$ learnt the secret of all the initial $n-1$ agents, we obtain the total of $n-1$ calls required by Lemma 6.11 for the secret of each of the initial $n-1$ agents to be learnt by the other agents in the scenario. From the foregoing we see therefore that $f(n)$ is given by the recursion:

$$f(n) = f(n-1) + (n-1) \tag{6.10}$$

To solve Equation 6.10 we proceed as follows.

$$\begin{aligned}
f(n) &= f(n-1) + (n-1) \\
&= f(n-2) + (n-2) + (n-1) \\
&\vdots \quad \vdots \\
&= f(n-k) + (n-k) + (n-(k-1)) + (n-(k-2)) + \cdots + (n-1) \\
&\vdots \quad \vdots \\
&= f(n-(n-1)) + (n-(n-1)) + (n-(n-2)) + \cdots + (n-1) \\
&= f(1) + 1 + 2 + \cdots + (n-1) \\
&= f(1) + \sum_{i=1}^{n-1} i \\
&= 0 + \tfrac{n(n-1)}{2} \\
&= \tfrac{n(n-1)}{2}
\end{aligned}$$

□

**Proposition 6.13.** *The maximum length of any execution sequence of a regular and successful epistemic gossip protocol is* $n(n-1)/2$.

*Proof.* From Definitions 4.19, 4.20 and 6.3, any execution sequence of a regular and successful epistemic gossip protocol is also regular and successful. Therefore the proof of the proposition follows from Proposition 6.12. □

Let us now continue with properties and proofs regarding the complete topology network, and then go on to those for the line, tree and circle topology networks, in turn.

## 6.2 Complete Topology Network

In this section, we consider a gossip scenario in which the network of agents is a complete topology network of $n$ agents. Let the arrangement of the agents be as shown in Figure

6.1, where $a_k$ and $a_j$ are any nodes such that $1 < j < k < n$. We present and prove some of the properties of Learn New Secrets, Known Information Growth de Dicto, Known Information Growth de Re, Possible Information Growth de Dicto, and Possible Information Growth de Re, on the complete topology network.

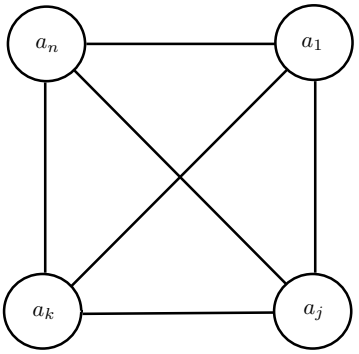



FIGURE 6.1: **Complete Topology Network.**

**Proposition 6.14.** *Learn New Secrets, Known Information Growth de Dicto and Known Information Growth de Re are terminating on a complete topology network.*

*Proof.* Take any of the three given protocols, and call it $\Pi$. Let the gossip tree for $\Pi$ be $\mathcal{T}_g$. Take any execution sequence $\sigma$ of $\Pi$. Suppose, towards contradiction, that $\sigma$ is infinite. Let $\sigma'$ be any prefix of $\sigma$. Now consider the set $S = \{(a_i, A_k) \mid \ a_i, a_k \in \mathsf{Ag}$ and $\mathcal{T}_g, \sigma' \models \neg K w_{a_i} A_k\}$. Clearly, for $\sigma' = e$ (that is, the empty history) $|S| = n(n-1)$, where $n = |\mathsf{Ag}|$. Also,

$$\mathcal{T}_g, \sigma' \models \varphi_\Pi(a_i, a_j) \quad \text{implies} \quad S \neq \emptyset \tag{6.11}$$

where $\varphi_\Pi(a_i, a_j)$ is the epistemic calling condition for $a_i$ to call $a_j$ for protocol $\Pi$, and $a_i, a_j \in \mathsf{Ag}$ (see Observation 6.8 (expression (6.6))).

As follows from Proposition 6.9, every call in an execution sequence of $\Pi$ removes at least one member from $S$. So there must be a finite number of calls after which $S = \emptyset$. And by contraposition on (6.11) we see that at such a finite number of calls, there is a prefix $\sigma'$ of $\sigma$ such that $\mathcal{T}_g, \sigma' \not\models \varphi_\Pi(a_i, a_j)$, and so no more calls are possible at such $\sigma'$. But this contradicts the assumption that the execution sequence $\sigma$ (of which $\sigma'$ is a prefix) is infinite. Therefore we conclude that $\sigma$ must be finite. □

**Proposition 6.15.** *Learn New Secrets, Known Information Growth de Dicto and Known Information Growth de Re are successful in a gossip scenario in which the network of agents is a complete topology network.*

*Proof.* We already know from Proposition 6.14 that all three given protocols are terminating. So suppose one of them is not successful, that is, it allows for a call sequence $\sigma = \sigma_1; \ldots; \sigma_k$ such that at $\sigma_k$ no calling condition is true, while at the same time not all agents know all secrets, that is, for some agents $a_i$ and $a_j$, we have $\mathcal{T}_g, \sigma_k \models \neg K w_{a_i} A_j$,

where $\mathcal{T}_g$ is the gossip tree of the supposed unsuccessful protocol. But it follows from (6.7) and (6.8) under Observation 6.8 that the calling conditions for $a_i$ to call $a_j$ are true at $\sigma_k$, for each of the protocols, which gives a contradiction. Hence, all call sequences for each of the three protocols are successful, that is, the protocols are successful. □

**Proposition 6.16.** *Possible Information Growth de Re is non-terminating in a gossip scenario in which the network of agents is a complete topology network and the size n of set of agents is greater than three.*

*Proof.* Let the set of agents be $Ag = \{a_1, a_2, \ldots, a_n\}$. Consider the following execution sequence $\sigma$:

$$\sigma \quad = \quad a_1a_2; a_3a_4; a_1a_2; a_3a_4; a_1a_2; \ldots$$

The execution sequence $\sigma$ is in the extension of Possible Information Growth protocol for any complete topology network of agents where $|Ag| > 3$. To demonstrate this, let us consider the calls in $\sigma$. In the first call, agent $a_1$ knows (and therefore considers it possible) that it would learn secret $A_2$ in the first call. And likewise, in the second call, agent $a_3$ knows that it will learn secret $A_4$ in the second call with agent $a_4$. In the third call, agent $a_1$ considers it possible that agent $a_2$ learnt some secret in the second call (in a call with agent $a_3$, for instance), and so agent $a_1$ calls agent $a_2$ in the third call, which proves redundant. Likewise in the fourth call, agent $a_3$ considers it possible that agent $a_4$ learnt some secret in the third call (in a call with agent $a_2$, for instance) and therefore agent $a_3$ calls agent $a_4$ in the fourth call which also proves redundant. Once again agent $a_1$ considers it possible that agent $a_2$ learnt some new secret in the preceding call, and thus for the execution sequence $\sigma$, the loop $\ldots; a_1a_2; a_3a_4; a_1a_2; a_3a_4; \ldots$ goes on infinitely. □

**Proposition 6.17.** *Possible Information Growth de Dicto is non-terminating in a gossip scenario in which the network of agents is a complete topology network and the size n of the set of agents is greater than three.*

*Proof.* The proposition follows from the proof of Proposition 6.16, and from the proof of Proposition 3.27 (and Observation 4.26) which shows that the extension of is equal to the extension of Possible Information Growth de Dicto. □

### 6.2.1 Synthesising a Maximum Length Successful Execution Sequence on a Complete Topology Network

In this subsection we present an example of a procedure for synthesising a maximum length successful execution sequence of some regular protocol. We show that the synthesised execution sequence is of length $n(n-1)/2$, where $n$ is the number of agents in the gossip scenario. We also show that the synthesised execution sequence is successful and that it is an execution sequence of Known Information Growth de Re and Known Information Growth de Dicto.

**Procedure 6.1** Longest sequence procedure for a complete topology network.

```
1: Given a complete topology network, as shown in Figure 6.1.
2: Let the agents be at the initial state; let n = |Ag|.
3:  for i = 2 to n
4:        Let a_1 call a_i
5:        for j = i − 1 to 2
6:              Let a_1 call a_j
7:        end for
8:  end for
```

Now, consider Procedure 6.1. It generates a single execution sequence $\sigma$. Every call in $\sigma$ is initiated by the agent $a_1$. For $n = 4$, the generated execution sequence is

$$\sigma = a_1a_2; a_1a_3; a_1a_2; a_1a_4; a_1a_3; a_1a_2$$

For $n > 1$ agents, how long is the execution sequence $\sigma$ generated by Procedure 6.1? We answer as follows.

**Proposition 6.18.** *The length of the execution sequence, $\sigma$, generated by Procedure 6.1 is given by $n(n-1)/2$, where $n$ is the number of agents in the gossip scenario.*

*Proof.* Consider the loop which begins at Line 3. This loop iterates for $n-1$ times, so Line 4 generates a total of $n-1$ calls. For the $k^{th}$ iteration of the outer loop, the inner loop executes for $k-1$ times, where $1 \leq k \leq n-1$. So the total number of calls generated by Line 6 is given by the summation:

$$0 + 1 + \cdots + n - 2 = \sum_{m=0}^{n-2} m = \frac{(n-1)(n-2)}{2}$$

Therefore the total number of calls generated by Procedure 6.1, and hence the length of $\sigma$, is given by:

$$(n-1) + \frac{(n-1)(n-2)}{2} = \frac{n(n-1)}{2}$$ □

**Lemma 6.19.** *The number of fresh calls generated by Procedure 6.1 is exactly $n-1$.*

*Proof.* From the 'for' loop at Line 3 of Procedure 6.1, consider any $2 < i \leq n$. Notice that prior to the $a_1a_i$ call generated at Line 4, there were $a_1a_k$ calls for all $2 \leq k < i$, generated at previous iterations of this 'for' loop. But the calls generated at Line 6 are calls $a_1a_j$, for all $2 \leq j < i$. So we see that all the calls generated at Line 6 must have been generated previously in an earlier iteration of the 'for' loop at Line 3, and so these calls are not fresh calls. But this implies that all the fresh calls are generated at Line 3, since calls generated at that point always precede those generated at Line 6. Furthermore, since $a_1$ always initiates the call to $a_i$ at Line 3, and the value of $i$ increases by one for each iteration of the loop, we see that agent $a_i$ is indeed a fresh caller in each of the calls generated in that line. And since there are exactly $n-1$ such values of $i$ at Line 3, we conclude that there are exactly $n-1$ fresh calls in Procedure 6.1. □

**Proposition 6.20.** *For a gossip scenario with n agents, the call sequence generated by Procedure 6.1 is an execution sequence of Known Information Growth de Re.*

*Proof.* Recall that the calling condition for Known Information Growth de Re states that for any agent $a_i$ to call another agent $a_j$, there must be a secret $A_k$ such that $a_i$ knows that one of $a_i$ and $a_j$ will learn $A_k$ from the $a_ia_j$ call; moreover, agent $a_i$ and $a_j$ must be neighbours on the network graph.

Consider all the calls generated at Line 4 of Procedure 6.1. From Lemma 6.19 we see that all such calls are fresh calls, and that agent $a_i$ is a fresh caller in such calls. So agent $a_1$ does not know the secret of such $a_i$, and knows that it will learn the secret of agent $a_i$ in each of those $a_1a_i$ calls at Line 4. Thus the epistemic calling condition for Known Information Growth de Re is satisfied for all calls generated at Line 4.

At Line 6, consider any $i$ such that $2 \leq i \leq n$. After generating the call $a_1a_i$ at Line 4, we see that all the calls generated at Line 6 are calls from $a_1$ to $a_j$ where $1 < j < i$. Observe that all values of $j$ considered at Line 6 are such that $a_1$ had called $a_j$ previously in the execution sequence. But $a_1$ is involved in all the calls so far, and since $a_i$ was a fresh caller in the $a_1a_i$ call generated at Line 4, agent $a_1$ knows that all the previous agents it has called so far, namely, $a_j$, $(1 < j < i)$, have not learnt the secret of agent $a_i$. And so for the $a_1a_j$ calls generated at Line 6, agent $a_1$ knows that $a_j$ will learn the secret of agent $a_i$ in such a call. Thus the epistemic calling condition for Known Information Growth de Re is again satisfied for all the calls generated at Line 6. □

**Proposition 6.21.** *The call sequence generated by Procedure 6.1 is successful.*

*Proof.* In Procedure 6.1 we are given that the calls begin at the initial state. To show that the generated execution sequence is successful we have to show that it is finite, and that it ends in a goal state.

Consider the 'for' loop at Lines 5-7 of Procedure 6.1. For any $2 \leq i \leq n$, let the set $\mathsf{Ag}'$ be defined as $\mathsf{Ag}' = \{a_j |\ i - 1 \geq j \geq 2\}$. So we see that Line 6 generates a sequence of calls in which agent $a_1$ calls each of the agents in $\mathsf{Ag}'$, for a given $i$. Therefore after that 'for' loop, each of the agents in $\mathsf{Ag}'$ learns all the secrets that agent $a_1$ knows after the call generated at Line 4 for the given $i$. Moreover the calls generated at Line 4 ensure that agent $a_1$ calls each of all the agents from $a_i = a_2$ to $a_n$, in order of increasing value of $i$. This implies that at $i = n - 1$, agent $a_1$ must have learnt the secrets of the agents $a_2$ to $a_{n-1}$, due to the calls generated at Line 4. So for $i = n$, the $a_1a_n$ call generated at Line 4 ensures that both $a_1$ and $a_n$ know all the secrets in the scenario. Finally, the Loop at Lines 5-7 ensures that for $i = n$, all agents in $\mathsf{Ag}'$, namely $a_2$ to $a_{n-1}$ learn all the secrets in the scenario, thus we obtain a goal state.

To see that the generated execution sequence is finite, consider that the outer loop (i.e. the 'for' loop beginning at Line 3) iterates for $n - 1$ times, and for the $k^{th}$ iteration of the outer loop, the inner loop (i.e. the 'for' loop beginning at Line 5 ) iterates for $k - 1$ times. Since the value of $n$ is assumed to be finite for the gossip scenario, it is clear that Procedure 6.1 terminates, and so the generated execution sequence is finite. □

**Lemma 6.22.** *In every call generated at Line 6 of Procedure 6.1 the total number of secrets learnt by the calling pair is exactly one.*

*Proof.* Consider line 4 of Procedure 6.1. For any given $2 \leq i \leq n$, let the set $\mathsf{Ag}'$ be defined as $\mathsf{Ag}' = \{a_j |\ 1 \leq j < i\}$. We first show that before every execution of Line 4, for any $2 \leq i \leq n$, every agent in $\mathsf{Ag}'$ knows only the secret of every agent in $\mathsf{Ag}'$; in the following we prove this property and by it show that in every call generated at Line 6 the total number of secrets learnt by the calling pair is exactly one. We proceed by induction over $i$, as follows.

**Base Case**

Initially, at $i = 2$, $\mathsf{Ag}' = \{a_1\}$, so indeed before the call generated at Line 4, every agent in $\mathsf{Ag}'$ knows only the secret of every agent in $\mathsf{Ag}'$.

**Inductive Hypothesis**

As the inductive hypothesis, for any $3 \leq k \leq n$, suppose that for $i = k - 1$ we have that every agent in $\mathsf{Ag}'$ knows only the secret of every agent in $\mathsf{Ag}'$, before the call generated at Line 4, at the loop iteration corresponding to the value of $i$.

**Inductive Step**

Now let $i = k$. At Line 4, the call $a_1 a_k$ is generated. From the proof of Lemma 6.19 we know that this call is a fresh call, and that $a_k$ is a fresh caller in the call. Based on the inductive hypothesis, we see that after this $a_1 a_k$ call, agent $a_1$ knows one more secret, namely, that of agent $a_k$, than the other agents in $\mathsf{Ag}'$. Since the 'for' loop at Lines 5-7 ensures that a sequence of calls is generated where $a_1$ calls each of all the other agents in $\mathsf{Ag}'$, we have that in each of those calls the only secret that is learnt is that of $a_k$. Thus we see that over all values of $k$, we have that all the calls generated at Line 6 preserve the property that the total number of secrets learnt by the calling pair is exactly one. □

**Proposition 6.23.** *Let* $\Pi$ *be a regular gossip protocol, and let* $\Pi$ *be successful. Then, on a complete topology network, no execution sequence of* $\Pi$ *is longer than the execution sequence generated by Procedure 6.1.*

*Proof.* Let the call sequence generated by Procedure 6.1 be $\sigma$. By Lemma 6.10 we know that $\sigma$ should have at most $n - 1$ fresh calls, since $\sigma$ is successful (from Proposition 6.21). But we see from Lemma 6.19 that $\sigma$ has exactly $n - 1$ fresh calls, and that Line 4 generates only fresh calls *and* all the fresh calls. Furthermore, the other line in Procedure 6.1 where calls are generated is at Line 6, and from Lemma 6.22 we know that in each of those calls the total number of secrets learnt in the call is exactly one. Clearly, it is not possible to obtain another execution sequence of any regular and successful protocol that is longer than $\sigma$, unless such an execution sequence contains some call in which the total number of secrets learnt is zero, which then contradicts the assumption that the protocol is regular. □

**Corollary 6.24.** *Given a complete topology network of agents, the maximum length of an execution sequence in the extension of a regular and successful epistemic gossip protocol is* $n(n-1)/2$.

*Proof.* The proof follows from the proof of Proposition 6.23 in which we show that the length of the execution sequence generated by Procedure 6.1 is the maximum for any execution sequence of a regular and successful gossip protocol on a complete topology network. □

**Corollary 6.25.** *Given a complete topology network of agents, the maximum length of an execution sequence of Known Information Growth de Re and Known Information Growth de Dicto is* $n(n-1)/2$.

*Proof.* From Corollary 6.24 we see that the length of the execution sequence generated by Procedure 6.1 is the maximum for any regular and successful gossip protocol in the complete topology network. In particular, the length of the execution sequence generated by Procedure 6.1 is the maximum for Known Information Growth de Re (from Proposition 6.20). Finally, the length of the execution sequence generated by Procedure 6.1 is the maximum for Known Information Growth de Dicto (from Proposition 3.27 and Observation 4.26: every execution sequence of Known Information Growth de Re is an execution sequence of Known Information Growth de Dicto). □

Note that the results in Proposition 6.23, Corollary 6.24 and Corollary 6.25 all agree with the more general results in Proposition 6.12 and Proposition 6.13.

### 6.2.2 Synthesising Minimum Length Successful Execution Sequences on a Complete Topology Network

We now consider the minimum length of a successful execution sequence of Learn New Secrets, Known Information Growth de Dicto, Known Information Growth de Re, Possible Information Growth de Dicto and Possible Information Growth de Re. We then describe a procedure for synthesising some minimum length successful execution sequences for the foregoing protocols on a complete topology network.

**Proposition 6.26.** *Given a complete topology network of agents in a gossip scenario, the minimum length of a successful execution sequence is* $2n-4$ *for Known Information Growth de Dicto (Protocol* 2*), Known Information Growth de Re (Protocol* 3*), Possible Information Growth de Dicto (Protocol* 4*) and Possible Information Growth de Re (Protocol* 5*), where* $n = |\mathsf{Ag}|$.

*Proof.* In Section 1.1, we showed that the Fixed Schedule yields execution sequences of length $2n-4$. In (the proof of) Proposition 3.27 (and Observation 4.26) we showed that the extension of the Fixed Schedule is a subset of the extension of Protocol 3. We know that less than $2n-4$ calls are insufficient to distribute all secrets as shown, for

example, in [63]. Therefore the length of the shortest successful execution sequence in the extension of Protocol 3 is $2n - 4$.

Moreover, from Proposition 3.27 (and Observation 4.26), we also know that: the extension of Protocol 3 is a subset of the extension of Protocol 2, and the extension of Protocol 2 is a subset of the extension of Protocol 5, which is equal to the extension of Protocol 4. Therefore we conclude that the length of the shortest successful execution sequence of Protocol 2, 5 and 4 is also $2n - 4$. □

Now we come to the synthesis of minimum length execution sequences.

*Procedure* 5.1. Consider a set $\mathsf{Ag} = \{a_1, \ldots, a_n\}$ of agents on a complete topology network such that $|\mathsf{Ag}| > 4$. Designate four of the agents, say $a_1, a_2, a_3, a_4$. Further designate one of the four designated agents, say $a_4$. Now consider a 4-cycle $\omega$ on the network graph that connects all four designated agents, and consider a path $\pi$ on the network graph that begins on any of the non-designated nodes, and passes through each of the other non-designated nodes exactly once, and then terminates on the node $a_4$. See Figure 6.2 for a sample illustration of $\omega$ and $\pi$ in complete topology network consisting of $n$ agents.

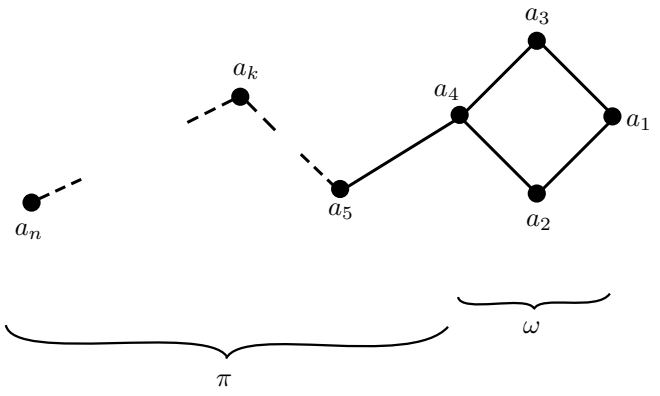



FIGURE 6.2: **Highlighting some edges of a complete topology network.**

This procedure then executes the following three steps in the order Step 1, Step 2 and Step 3:

Step 1 Let the non-designated agents make calls as follows:

$$a_n a_{n-1}; \ldots; a_k a_{k-1}; \ldots; a_5 a_4, \quad \text{where } n > k > 5$$

Step 2 Let the designated agents call each other according to the *Four-Agent Protocol* as follows: any two agents make the first call; the second call is then between the remaining two agents; the third call is then between an agent who made the first call and an agent who made the second call; and the fourth call is between the two who were not chosen in the third call.

Step 3 Let each of the non-designated agents make one call with any one of the designated agents apart from agent $a_4$.

*Observation* 6.27. Notice that after Step 1, agent $a_4$ knows the secret of all the non-designated agents (beginning with agent $a_n$, each agent $a_k$ calls agent $a_{k-1}$, up to the call $a_5a_4$, at which agent $a_4$ learns the secrets of all the non-designated agents). Next, Step 2 is such that all the four designated agents know all the secrets of each other after the step. So, since agent $a_4$ knows the secret of all the non-designated agents at the end of Step 1, and the calling agents exchange all the secrets they know in each call, it follows that at the end of the Step 2 all the designated agents $(a, b, c, d)$ know all the secrets of all agents in the scenario. Finally in Step 3, each of the $n-4$ non-designated agents calls any one of the designated agents apart from agent $a_4$. So clearly, after the calls in Step 3, all the agents know all the secrets in the scenario. And so we see that the execution sequences of Procedure 5.1 are successful.

Notice also that Step 1 of Procedure 5.1 requires $n-4$ calls (there are $n-3$ agents involved in this step, these agents are on the path $\pi$, and each edge of the path $\pi$ is used only once, where a use of an edge denotes a call between adjoining nodes). Obviously Step 2 requires four calls. Step 3 requires $n-4$ calls, one for each of the $n-4$ non-designated agents. So, summing up, the three steps together yield execution sequences of length $2n-4$, which corresponds to the shortest successful execution sequence length to solve the gossip problem, as proved by, for example, [63].

**Proposition 6.28.** *Given a complete topology network of agents in a gossip scenario, all the execution sequences of Procedure 5.1 are successful execution sequences of Learn New Secrets, Known Information Growth de Re, Known Information Growth de Dicto, Possible Information Growth de Re and Possible Information Growth de Dicto.*

*Proof.* Observe that in each call of each step of Procedure 5.1, the agent who initiates the call does not know the secret of the other caller prior to the call. This implies that for every execution sequence $\sigma$ of Procedure 5.1, the epistemic calling condition for Learn New Secrets is satisfied for each call of $\sigma$. Therefore such execution sequence $\sigma$ is an execution sequence of Learn New Secrets (see Definition 4.13). Furthermore, from Observation 6.27, we see that all such $\sigma$ are successful.

Moreover, from Proposition 3.27 (and Observation 4.26), we know that: the extension of Learn New Secrets is a subset of the extension of Known Information Growth de Re, whose extension is in turn a subset of the extension of Known Information Growth de Dicto, whose extension is in turn a subset of the extension of Possible Information Growth de Re, whose extension is equal to the extension of Possible Information Growth de Dicto. Therefore we conclude that every execution sequence $\sigma$ of Procedure 5.1, is also a successful execution sequence of Known Information Growth de Dicto, Known Information Growth de Re, Possible Information Growth de Dicto and Possible Information Growth de Re. □

**Proposition 6.29.** *Given a complete topology network of agents in a gossip scenario, the minimum length of a successful execution sequence of Learn New Secrets is* $2n-4$.

*Proof.* Consider an example execution sequence of Procedure 5.1, as follows:

$$\sigma = a_n a_{n-1}; \ldots; a_k a_{k-1}; \ldots; a_5 a_4; a_4 a_3; a_2 a_1; a_4 a_2; a_3 a_1; a_n a_3; \ldots; a_k a_3; \ldots; a_5 a_3$$

where $n > k > 5$.

From Proposition 6.28 we know that $\sigma$ is a successful execution sequence of Learn New Secrets. Also, the length of $\sigma$ is given by $2n - 4$ (see Observation 6.27). Since it is not possible to obtain a successful execution sequence that is less than $2n - 4$ in length (see [63]), we conclude that the length of the shortest successful execution sequence of Learn New Secrets is $2n - 4$. □

From Proposition 6.29, and from the last lines of the proof of Proposition 6.28 we can also conclude that on a complete topology network, the shortest successful execution sequence length is given by $2n - 4$ for Known Information Growth de Re, Known Information Growth de Dicto, Possible Information Growth de Re and Possible Information Growth de Dicto, since we see that an execution sequence of Learn New Secrets is also an execution sequence of the foregoing protocols.

Our results from Propositions 6.26 and 6.29 show that our epistemic gossip protocols yield the same minimum successful execution sequence length given by non-epistemic procedures on the complete topology network, as proved in [13, 27, 61, 63].

## 6.3 Line Topology Network

In this section we consider gossip scenarios in which the network of agents is a line topology network of $n$ agents. Without loss of generality, assume that the nodes of each network are arranged as shown in Figure 6.3. We begin with the description of a notation that makes it easier for us to talk about some properties of gossip protocols on a line topology network, and then present and prove some epistemic properties of gossip scenarios on a line topology network. We then present and prove some of the properties of Learn New Secrets, Known Information Growth de Dicto, Known Information Growth de Re, Possible Information Growth de Dicto, and Possible Information Growth de Re, on a line topology network.

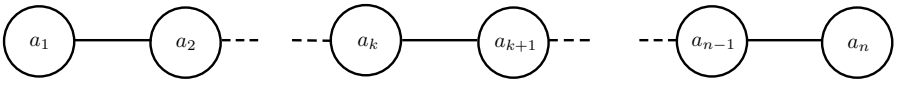



FIGURE 6.3: **Line Topology Network.**

**Definition 6.30** (Orientation)**.** Given a set of agents $Ag = \{a_1, \ldots, a_n\}$, and a corresponding set of secrets $S = \{A_1, \ldots, A_n\}$, let each agent $a_i \in \mathsf{Ag}$ correspond to a node

of a line topology network, as shown in Figure 6.3. Then,

$$\begin{array}{lcl} \overrightarrow{a_i} & = & \{a_j |\ i \leq j \leq n\} \\ \overrightarrow{a_i}^{m} & = & \{a_j |\ i \leq j \leq i+m\},\ \text{where } n \geq i+m \\ \overleftarrow{a_i} & = & \{a_j |\ i \geq j \geq 1\} \\ \overleftarrow{a_i}^{m} & = & \{a_j |\ i \geq j \geq i-m\},\ \text{where } 1 \leq i-m \end{array}$$

We refer to $\overleftarrow{a_i}$ and $\overrightarrow{a_i}$ as the *left* and the *right* orientation, respectively, of agent $a_i$ on a line network. Furthermore, let for any $Bg \subseteq \mathsf{Ag}$, and any call sequence $\sigma$:

$$S(Bg, \sigma) \quad = \quad \{A_j |\ \exists a_i \in Bg \text{ and } Kw_{a_i} A_j \text{ at } \sigma\}$$

That is, $S(Bg, \sigma)$ denotes all secrets that are known at $\sigma$ by any of the agents in $Bg$.

**Definition 6.31** (Intermediary and Terminus)**.** An intermediary is an agent $a_i$ such that $1 < i < n$. An agent $a_k$ is a terminus if it is not an intermediary.

Notice that on a line topology network with $|\mathsf{Ag}| > 2$, each call must involve at least one intermediary agent, since otherwise no two termini can call each other.

### 6.3.1 Epistemic Properties of Gossip Scenarios on Line Topology Network

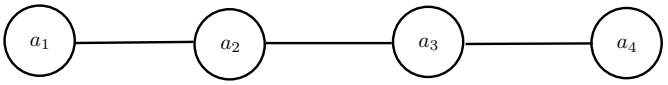



FIGURE 6.4: **An example of a line topology network.**

Now, consider the line topology network shown in Figure 6.4, and assume $n = 4$ agents $a_1, a_2, a_3$ and $a_4$. We see that agent $a_4$, for example, cannot learn the secret of agent $a_2$ unless through agent $a_3$. In other words, agent $a_3$ must learn the secret of agent $a_2$ before agent $a_4$ does, and likewise $a_2$ must learn secret of agent $a_4$ before agent $a_1$ learns it, etc. So we see that the line network imposes a specific set of constraints on how the secrets *flow* in the network over an execution sequence of calls. Furthermore, since we assume that the network topology is commonly known by the agents, it follows that agent $a_3$, for example, knows whether agent $a_2$ knows the secret of agent $a_4$, since agent $a_3$ knows that there is no other way agent $a_2$ can learn the secret of agent $a_4$ before (or apart from) agent $a_3$. Theorem 6.32 formally describes some basic properties and constraints of a gossip scenario set on a line topology network.

**Theorem 6.32.** *Let agents $a_1, \ldots, a_n$ be connected on a line topology network. Let $\mathcal{T}_g$ be any gossip tree, where $\sigma \in \mathcal{T}_g$ is an arbitrary execution sequence.*

*Let $i \leq j \leq k$. Then we have the following properties:*

$$\mathcal{T}_g, \sigma \models Kw_{a_i} A_k \rightarrow \bigwedge_{j \leq r \leq k} Kw_{a_j} A_r \tag{6.12}$$

$$\mathcal{T}_g, \sigma \models Kw_{a_i} A_k \leftrightarrow \bigwedge_{i \leq r \leq k} Kw_{a_i} A_r \tag{6.13}$$

$$\textit{Let } a_j \in \overset{\Rightarrow 1}{a_i}, \textit{ then } \mathcal{T}_g, \sigma \models Kw_{a_i} A_k \leftrightarrow K_{a_j} Kw_{a_i} A_k \tag{6.14}$$

$$\textit{Let } a_j \in \overset{\Rightarrow 1}{a_i}, \textit{ then } \mathcal{T}_g, \sigma \models \neg Kw_{a_i} A_k \leftrightarrow K_{a_j} \neg Kw_{a_i} A_k \tag{6.15}$$

*The properties* (6.12) *through* (6.15) *have also variants (6.12'), (6.13'), (6.14') and (6.15'), as follows, where $i \leq j \leq k$:*

$$\mathcal{T}_g, \sigma \models Kw_{a_k} A_i \rightarrow \bigwedge_{i \leq r \leq j} Kw_{a_j} A_r \tag{6.12'}$$

$$\mathcal{T}_g, \sigma \models Kw_{a_k} A_i \rightarrow \bigwedge_{i \leq r \leq k} Kw_{a_k} A_r \tag{6.13'}$$

$$\textit{Let } a_j \in \overset{\Leftarrow 1}{a_k}. \textit{ Then } \mathcal{T}_g, \sigma \models Kw_{a_k} A_i \leftrightarrow K_{a_j} Kw_{a_k} A_i \tag{6.14'}$$

$$\textit{Let } a_j \in \overset{\Leftarrow 1}{a_k}. \textit{ Then } \mathcal{T}_g, \sigma \models \neg Kw_{a_k} A_i \leftrightarrow K_{a_j} \neg Kw_{a_k} A_i \tag{6.15'}$$

Note that all eight properties (6.12) through (6.15) and (6.12') through (6.15') are common knowledge among all the agents in $\mathsf{Ag}$: notice that each of the properties is for an arbitrary $\sigma \in \mathcal{T}_g$, which means that each of the properties holds at every possible execution sequence in $\mathcal{T}_g$.

Property (6.12) captures the observation that if $a_i$ learns the secret $A_k$, then this secret must have travelled through $a_{k-1}, a_{k-2}, \ldots, a_j, \ldots$ all up to $a_i$. That is, given that $\sigma = \sigma_1, \ldots, \sigma_z$, then $a_{k-1}$ must learn secret $A_k$ through a call $\sigma_p$ with $a_k$, and agent $a_{k-2}$ must learn secret $A_k$ through a call $\sigma_q$ with $a_{k-1}$, (where $q > p$), $\ldots$, agent $a_j$ must learn secret $A_k$ through a call $\sigma_r$ with $a_{j+1}$ (where $r > q$), $\ldots$, and agent $a_i$ must learn secret $A_k$ through a call $\sigma_s$ with $a_{i+1}$ (where $s > r$). Furthermore, in each call the calling pair exchange all the secrets they know, thus beginning with $a_{k-1}$, each node $a_m$ through which $A_k$ travels, learns the unique secret of each of the nodes $a_{m'}$ where $m < m' \leq k$. Note that this property also holds in the extreme cases: where $i = j$ (in which case Property (6.12) becomes identical to Property (6.13)) and where $j = k$ (the consequent of Property (6.12) is true).

Property (6.14) depicts $a_j$'s knowledge of $a_i$'s knowledge, regarding some secret in the right orientation of agent $a_j$, where $a_i$ is at most one 'hop' to the left of $a_j$. Finally, Property (6.15) presents a variation of Property (6.14) where agent $a_j$ knows of the ignorance of agent $a_i$, regarding some secret in the right orientation of agent $a_j$.

Before we prove Theorem 6.32, let us first establish a few lemmas, as follows.

**Lemma 6.33.** *Given a gossip scenario on a line topology network, let* $\mathcal{T}_g = \langle H, R, F, \{Z_{a_j}\}\rangle$ *be any gossip tree, then there exists functions ml and mr such that, for each call sequence* $\sigma = \sigma_1; \sigma_2; \ldots$, *in* $H$, *and for each index* $p \geq 0$ *of call in* $\sigma$, *and for each agent* $a_m$, *we have*[*]*:*

1. $ml(\sigma, p, m) \leq m \leq mr(\sigma, p, m)$;

   *In words:* $a_{ml(\sigma,p,m)}$ *denotes the 'left-most' agent who knows* $a_m$*'s secret, and* $a_{mr(\sigma,p,m)}$ *denotes the 'right-most' agent who knows* $a_m$*'s secret.*

2. *for any agent* $a_j$ *and secret* $A_m$ *we have*

$$\mathcal{T}_g, \sigma \models Kw_{a_j} A_m \quad \textit{iff} \quad j \in [ml(\sigma, p, m), mr(\sigma, p, m)] \tag{6.16}$$

3. *after call* $\sigma_p$, *we have that*

   (a) *if* $ml(\sigma, p, m) = x > 1$, *then* $\mathcal{T}_g, \sigma \models Kw_{a_x} A_m \wedge K_{a_x} \neg Kw_{a_{x-1}} A_m$

   (b) *if* $mr(\sigma, p, m) = y < n$, *then* $\mathcal{T}_g, \sigma \models Kw_{a_y} A_m \wedge K_{a_y} \neg Kw_{a_{y+1}} A_m$

4. *after call* $\sigma_p$, *we have that*

   (a) *if* $ml(\sigma, p, m) = x \geq 1$, *then* $\mathcal{T}_g, \sigma \models Kw_{a_{x+1}} A_m \rightarrow K_{a_x} Kw_{a_{x+1}} A_m$

   (b) *if* $mr(\sigma, p, m) = y \leq n$, *then* $\mathcal{T}_g, \sigma \models Kw_{a_{y-1}} A_m \rightarrow K_{a_y} Kw_{a_{y-1}} A_m$

*Proof.*

- We prove Item 1 and Item 2 as follows:

  **Base Case** At the initial situation, $ml(\sigma, 0, m) = mr(\sigma, 0, m) = m$ (the only agent who knows $A_m$ is $a_m$). Clearly, Items 1 and 2 of Lemma 6.33 hold at the initial situation.

  **Inductive Hypothesis**

  Now for the inductive hypothesis, suppose $ml(\sigma, p-1, m)$ and $mr(\sigma, p-1, m)$ are given, for fixed $p - 1 > 0$ and for all $m$, and Items 1 and 2 hold after the call $\sigma_{p-1}$.

  **Inductive Step**

  For the inductive step, now consider call $\sigma_p$ in $\sigma = \sigma_1; \ldots; \sigma_i; \ldots; \sigma_p; \ldots$, without loss of generality we can assume that it is of the form $a_k a_{k+1}$ for some $k < n$. We adapt the two functions as follows:

$$ml(\sigma, p, m) \begin{cases} k & \text{if } ml(\sigma, p-1, m) = k+1 \\ ml(\sigma, p-1, m) & \text{else} \end{cases}$$

  and

$$mr(\sigma, p, m) \begin{cases} k+1 & \text{if } ml(\sigma, p-1, m) = k \\ mr(\sigma, p-1, m) & \text{else} \end{cases}$$

[*]For a call $\sigma_i$ in $\sigma$, the index of the call $\sigma_i$ is $i$.

In words, after the call $a_k a_{k+1}$, the left-most agent knowing $A_m$ becomes $a_k$ if it was $a_{k+1}$ before the call and stays unchanged otherwise, and the right-most agent knowing $A_m$ becomes $a_{k+1}$ if it was $a_k$ before the call, and stays unchanged otherwise. Clearly, this update of the function maintains the properties given in Items 1 and 2 of Lemma 6.33, for $\sigma_p$.

- We prove Item 3 as follows:

**Base Case**

For Item 3, at the initial situation, suppose $ml(\sigma, 0, m) = x > 1$ and $mr(\sigma, p, m) = y < n$, then also we have that $ml(\sigma, 0, m) = mr(\sigma, 0, m) = m$ also holds initially, and therefore the consequent of the property in Item 3(a) becomes $\mathcal{T}_g, \sigma \models Kw_{a_m} A_m \wedge K_{a_m} \neg Kw_{a_{m-1}} A_m$, and the consequent for Item 3(b) becomes $\mathcal{T}_g, \sigma \models Kw_{a_m} A_m \wedge K_{a_m} \neg Kw_{a_{m+1}} A_m$. Clearly $\mathcal{T}_g, \sigma \models Kw_{a_m} A_m$ at the initial situation. Also, $\mathcal{T}_g, \sigma \models K_{a_m} \neg Kw_{a_{m-1}} A_m$ (for Item 3(a)) and $\mathcal{T}_g, \sigma \models K_{a_m} \neg Kw_{a_{m+1}} A_m$ (for Item 3(b)) at the initial situation, since agent $a_m$, having not made any call yet, knows that none of its right hand and left hand neighbours knows the secret $A_m$.

**Inductive Hypothesis**

Now for the inductive hypothesis, suppose $ml(\sigma, p-1, m)$ and $mr(\sigma, p-1, m)$ are given, for fixed $p - 1 > 0$ and for all $m$, and Item 3 holds after the call $\sigma_{p-1}$.

**Inductive Step**

For the inductive step, now consider call $\sigma_p$ in $\sigma = \sigma_1; \ldots; \sigma_i; \ldots; \sigma_p; \ldots$, without loss of generality we can assume that it is of the form $a_k a_{k+1}$ for some $k < n$. We adapt the two functions as follows:

$$ml(\sigma, p, m) \begin{cases} k & \text{if } ml(\sigma, p-1, m) = k+1 \\ ml(\sigma, p-1, m) & \text{else} \end{cases}$$

and

$$mr(\sigma, p, m) \begin{cases} k+1 & \text{if } ml(\sigma, p-1, m) = k \\ mr(\sigma, p-1, m) & \text{else} \end{cases}$$

In words, after the call $a_k a_{k+1}$, the left-most agent knowing $A_m$ becomes $a_k$ if it was $a_{k+1}$ before the call and stays unchanged otherwise, and the right-most agent knowing $A_m$ becomes $a_{k+1}$ if it was $a_k$ before the call, and stays unchanged otherwise.

This update of the function maintains the properties given in Item 3(a) and Item 3(b) of Lemma 6.33, for $\sigma_p$. The argument for Item 3(a) is as follows. Suppose $ml(\sigma, p, m) = x > 1$. Then since $a_x$ is the left-most agent who knows the secret $A_m$, it must be the case that the agent $a_{x-1}$ does not know secret $A_m$. Suppose, towards contradiction, that at the call $\sigma_p$ it is the case

that $\neg K_{a_x} \neg K w_{a_{x-1}} A_m$ holds. That is, (see Definition 4.11), there is some execution sequence $\sigma'$ such that $\sigma \equiv_{a_x} \sigma'$, where $\sigma' = \sigma'_1; \ldots; \sigma'_i; \ldots; \sigma'_p; \ldots$ such that $i \leq p$, and $\sigma'_i = a_x a_{x-1}$ (or $\sigma'_i = a_{x-1} a_x$), and it is the case that after the call $\sigma'_i$, $K w_{a_{x-1}} A_m$ holds. Now since $\sigma' \in Z_{a_x}(\sigma)$ (see Definition 4.9), then it follows from the Definitions 4.7 and 4.9, that for all $\sigma'' \in Z_{a_x}(\sigma)$ it must be the case that $\sigma''_i$ is a call between agent $a_x$ and agent $a_{x-1}$, and after such call at $\sigma''_i$, agent $a_{x-1}$ knows the secret $A_m$. But from the semantics of $K_{a_x}$ operator given in Definition 4.11 this is equivalent to saying that $K_{a_x} K w_{a_{x-1}} A_m$ holds at $\sigma_p$, which, by veridicality, implies that $K w_{a_{x-1}} A_m$ holds at $\sigma_p$, and which then contradicts our earlier assumption that $a_x$ is the leftmost agent who knows secret $A_m$. Therefore we conclude that at $\sigma_p$, it is the case that $K_{a_x} \neg K w_{a_{x-1}} A_m$ holds.

The argument for Item 3(b) is analogous to the foregoing argument for Item 3(a).

- We prove Item 4 as follows:

**Base Case**

Finally, for Item 4 at the initial situation, suppose $ml(\sigma, 0, m) = x \geq 1$ and $mr(\sigma, p, m) = y \leq n$, then also we have that $ml(\sigma, 0, m) = mr(\sigma, 0, m) = m$ holds initially since the antecedents, namely, $\mathcal{T}_g, \sigma \models K w_{a_{m+1}} A_m$ and $\mathcal{T}_g, \sigma \models K w_{a_{m-1}} A_m$ are false at the initial situation.

**Inductive Hypothesis**

Now for the inductive hypothesis, suppose $ml(\sigma, p-1, m)$ and $mr(\sigma, p-1, m)$ are given, for fixed $p - 1 > 0$ and for all $m$, and Item 4 holds after the call $\sigma_{p-1}$.

**Inductive Step**

For the inductive step, now consider call $\sigma_p$ in $\sigma = \sigma_1; \ldots; \sigma_i; \ldots; \sigma_p; \ldots$, without loss of generality we can assume that it is of the form $a_k a_{k+1}$ for some $k < n$. We adapt the two functions as follows:

$$ml(\sigma, p, m) \begin{cases} k & \text{if } ml(\sigma, p-1, m) = k+1 \\ ml(\sigma, p-1, m) & \text{else} \end{cases}$$

and

$$mr(\sigma, p, m) \begin{cases} k+1 & \text{if } ml(\sigma, p-1, m) = k \\ mr(\sigma, p-1, m) & \text{else} \end{cases}$$

In words, after the call $a_k a_{k+1}$, the left-most agent knowing $A_m$ becomes $a_k$ if it was $a_{k+1}$ before the call and stays unchanged otherwise, and the right-most agent knowing $A_m$ becomes $a_{k+1}$ if it was $a_k$ before the call, and stays unchanged otherwise.

This update of the function maintains the properties given in Item 4(a) and Item 4(b) of Lemma 6.33, for $\sigma_p$. The argument for Item 4(a) is as follows.

Suppose $ml(\sigma, p, m) = x \geq 1$. Suppose also that $\mathcal{T}_g, \sigma \models Kw_{a_{x+1}} A_m$ also holds. Then $\sigma$ is of the form $\sigma = \sigma_1; \ldots; \sigma_i; \ldots; \sigma_p; \ldots$, where $i \leq p$, and $\sigma_i = a_x a_{x+1}$ (or $\sigma_i = a_{x+1} a_x$), and, either $a_x$ learnt secret $A_m$ from $a_{x+1}$ or (in the case where $x = m$) $a_{x+1}$ learnt $A_m$ from $a_x$, at the call $\sigma_i$ (note that here, given the constraints of the line topology network, it is not possible to have an execution sequence in which, where $x \neq m$, we have that $a_x$ learnt secret $A_m$ from any other agent other than $a_{x+1}$). And from Definitions 4.7 and 4.9, we see that for all $\sigma' \in Z_{a_x}(\sigma)$, it is the case that $\sigma' = \sigma'_1; \ldots; \sigma'_i; \ldots; \sigma'_p; \ldots$, and $\sigma'_i \equiv_{a_x} \sigma_i$, that is the same agents, namely, $a_x$ and $a_{x+1}$, called each other at the $i^{th}$ call in both $\sigma$ and $\sigma'$, *and*, the set of secrets known by $a_x$ and $a_{x+1}$ at $\sigma_i$ are the same, and are equal to the set of secrets known by $a_x$ and $a_{x+1}$ at $\sigma'_i$. Therefore, for all $\sigma' \in Z_{a_x}(\sigma)$, it is the case that both $a_x$ and $a_{x+1}$ know secret $A_m$ at $\sigma'_i$. But from the semantics of '$K_{a_x}$' operator given in Definition 4.11 this means that $K_{a_x} Kw_{a_{x+1}} A_m$ holds at $\sigma$. That is, $\mathcal{T}_g, \sigma \models K_{a_x} Kw_{a_{x+1}} A_m$ And given our foregoing assumptions we conclude that $\mathcal{T}_g, \sigma \models Kw_{a_{x+1}} A_m \rightarrow K_{a_x} Kw_{a_{x+1}} A_m$.

The argument for Item 4(b) is analogous to the foregoing argument for Item 4(a). □

**Lemma 6.34.** *Given a gossip scenario on a line topology network, let $i \leq j \leq k$ on the line topology network. Let $\mathcal{T}_g$ be any gossip tree, and let $\sigma = \sigma_1; \sigma_2; \ldots$ be an execution sequence in $\mathcal{T}_g$. Let the functions ml and mr be as described in Lemma 6.33. Then:*

$$\mathcal{T}_g, \sigma \models \bigwedge_{ml(\sigma,p,k) \leq x \leq k} Kw_{a_{ml(\sigma,p,k)}} A_x \tag{6.17}$$

*and*

$$\mathcal{T}_g, \sigma \models \bigwedge_{mr(\sigma,p,i) \geq x \geq i} Kw_{a_{mr(\sigma,p,i)}} A_x \tag{6.18}$$

In words, Property (6.17) states that at $\sigma$ it is the case that the left-most agent who knows secret $A_k$ at call index $p$, also knows all the unique secrets of all the agents on its right orientation (including its own secret), right up to the unique secret of agent $a_k$. Similarly, Property (6.18) states that at $\sigma$ it is the case that the right-most agent who knows secret $A_i$ at call index $p$, also knows all the unique secrets of all the agents on its left orientation (including its own secret), right up to the unique secret of agent $a_i$.

*Proof.* Consider the Property (6.17) under Lemma 6.34. We proceed by induction over the index $p$ of call in $\sigma$.

**Base Case**

At the initial situation, $ml(\sigma, 0, k) = mr(\sigma, 0, k) = k$, so Property (6.17) is obviously true.

**Inductive Hypothesis**

Now, as the inductive hypothesis, suppose $ml(\sigma, p-1, k)$ is given, for fixed $p-1 > 0$ and for all $k$, and Property (6.17) holds after the call $\sigma_{p-1}$ of $\sigma$. That is,

$$\mathcal{T}_g, \sigma \models \bigwedge_{ml(\sigma,p-1,k) \leq x \leq k} Kw_{a_{ml(\sigma,p-1,k)}} A_x \tag{6.19}$$

**Inductive Step**

For the inductive step, consider call $\sigma_p$, and without loss of generality, assume that it is of the form $a_j a_{j+1}$ for some $j < k$. We adapt the function $ml$ as follows:

$$ml(\sigma, p, k) \begin{cases} j & \text{if } ml(\sigma, p-1, m) = j+1 \\ ml(\sigma, p-1, k) & \text{else} \end{cases}$$

That is, after the call $a_j a_{j+1}$, the left-most agent knowing $A_k$ becomes $a_j$ if it was $a_{j+1}$ before the call, and stays unchanged otherwise. But during such $a_j a_{j+1}$ call at $\sigma_p$, we have that both agents exchange all the secrets they currently know. That is, (see Definition 6.30), $S(\{a_j\}, \sigma'_p) = S(\{a_{j+1}\}, \sigma'_p) = S(\{a_j\}, \sigma'_{p-1}) \cup S(\{a_{j+1}\}, \sigma'_{p-1})$, where $\sigma'_p = \sigma_1; \ldots; \sigma_p$ is the prefix of length $p$ of $\sigma$, comprising of the first $p$ calls in $\sigma$. Therefore it must be the case that after the $a_j a_{j+1}$ call at $\sigma_p$, agent $a_j$ knows all the secrets that $a_{j+1}$ knew at $\sigma_{p-1}$, which from the inductive hypothesis comprises the unique secret of agents between $a_{j+1}$ and $a_k$ inclusive. So we conclude that after the call $\sigma_p$, Property (6.17) is maintained.

The argument for the property in (6.18) is analogous to that given in the foregoing for Property (6.17). □

## Proof of Theorem 6.32

We now present proofs for the properties given in Theorem 6.32.

*Proof of Property* (6.12). Let $\sigma = \sigma_1; \ldots; \sigma_p$, and let $ml$ and $mr$ be as described in Lemma 6.33. Suppose $\mathcal{T}_g, \sigma \models Kw_{a_i} A_k$ holds. Then from (6.16) we see that $a_i \in [ml(\sigma, p, k), mr(\sigma, p, k)]$. Since $i \leq j \leq k$, and from the proof of Lemma 6.33, we see that there is some index $0 \leq s \leq p$, such that $ml(\sigma, s, k) = j$. So then from Lemma 6.34 we conclude that:

$$\mathcal{T}_g, \sigma \models \bigwedge_{ml(\sigma,s,k) \leq r \leq k} Kw_{a_j} A_r$$

That is,

$$\mathcal{T}_g, \sigma \models \bigwedge_{j \leq r \leq k} Kw_{a_j} A_r$$

□

*Proof of Property* (6.13). Let $\sigma = \sigma_1; \ldots; \sigma_p$, and let $ml$ and $mr$ be as described in Lemma 6.33.

**'$\Longrightarrow$'-direction** This direction is the special case of Property (6.12) where $i = j$.

**'$\Longleftarrow$'-direction** Suppose $\mathcal{T}_g, \sigma \models \bigwedge_{i \leq r \leq k} Kw_{a_i} A_r$ holds. Then, trivially from propositional logic, we have that $\mathcal{T}_g, \sigma \models Kw_{a_i} A_k$ holds. □

*Proof of Property* (6.14). Given that $a_j \in \overset{\Rightarrow 1}{a_i}$ and given that $i \leq j \leq k$, we distinguish the following cases.

**Case $i = j = k$.** Here we have to prove that:

$$\mathcal{T}_g, \sigma \models Kw_{a_i} A_i \leftrightarrow K_{a_i} Kw_{a_i} A_i$$

**'$\Longrightarrow$'-direction** Suppose $\mathcal{T}_g, \sigma \models Kw_{a_i} A_i$ holds. From Definition 4.7, if agent $a_i$ considers $\sigma$ and $\sigma'$ to be equivalent, then agent $a_i$ also knows the same secrets at both $\sigma$ and $\sigma'$; so if agent $a_i$ knows secret $A_i$ at $\sigma$, then it also knows secret $A_i$ at $\sigma'$. Therefore from Definitions 4.7 and 4.9, we see that for every $\sigma' \in Z_{a_i}(\sigma)$ we have that $\mathcal{T}_g, \sigma' \models Kw_{a_i} A_i$. And from the semantics of '$K_{a_i}$' operator, we have that $\mathcal{T}_g, \sigma \models K_{a_i} Kw_{a_i} A_i$.

**'$\Longleftarrow$'-direction** Suppose $\mathcal{T}_g, \sigma \models K_{a_i} Kw_{a_i} A_i$ holds. That is, for every $\sigma' \in Z_{a_i}(\sigma)$, we have that $\mathcal{T}_g, \sigma' \models Kw_{a_i} A_i$. But $\sigma \in Z_{a_i}(\sigma)$, therefore we conclude that $\mathcal{T}_g, \sigma \models Kw_{a_i} A_i$.

**Case $i < j = k$.** We have to prove that:

$$\mathcal{T}_g, \sigma \models Kw_{a_i} A_j \leftrightarrow K_{a_j} Kw_{a_i} A_j$$

**'$\Longrightarrow$'-direction** Suppose $\mathcal{T}_g, \sigma \models Kw_{a_i} A_j$ holds. Then $\sigma$ must be of the form $\sigma = \sigma_1; \ldots; \sigma_m; \ldots$, such that at $\sigma_m$, agent $a_i$ and agent $a_j$ called each other for the first time and exchanged all the secrets they knew at the call, including their own unique secrets (given the constraints of the line topology network, agent $a_i$ cannot learn secret $A_j$, other than through a call with agent $a_j$, and this fact is known by all the agents since the topology of the network is commonly known). Furthermore, from Definitions 4.7 and 4.9, for every $\sigma' \in Z_{a_j}(\sigma)$, we have that $\sigma' = \sigma'_1; \ldots; \sigma'_m; \ldots$ and $\sigma_m \equiv_{a_j} \sigma'_m$, and since agent $a_i$ knows $A_j$ at $\sigma_m$, then from Definition 4.7 we see that agent $a_i$ also knows secret $A_j$ at $\sigma'_m$, for every $\sigma' \in Z_{a_j}(\sigma)$. And from the semantics of '$K_{a_j}$' operator (see Definition 4.11), we conclude that $\mathcal{T}_g, \sigma \models K_{a_j} Kw_{a_i} A_j$.

**'$\Longleftarrow$'-direction** Suppose $\mathcal{T}_g, \sigma \models K_{a_j} Kw_{a_i} A_j$ holds. From the semantics of '$K_{a_j}$' operator (see Definition 4.11), this means that for every $\sigma' \in Z_{a_j}(\sigma)$, we have that $\mathcal{T}_g, \sigma' \models Kw_{a_i} A_j$. But $\sigma \in Z_{a_j}(\sigma)$, therefore we conclude that $\mathcal{T}_g, \sigma \models Kw_{a_i} A_j$.

**Case** $i = j < k$**.** We have to prove that:

$$\mathcal{T}_g, \sigma \models Kw_{a_i} A_k \leftrightarrow K_{a_i} Kw_{a_i} A_k$$

**'$\Longrightarrow$'-direction** Suppose $\mathcal{T}_g, \sigma \models Kw_{a_i} A_k$ holds. From Definition 4.7, if agent $a_i$ considers any executions sequences $\sigma$ and $\sigma'$ to be equivalent, then agent $a_i$ also knows the same secrets at both $\sigma$ and $\sigma'$; so if agent $a_i$ knows secret $A_k$ at $\sigma$, then it also knows secret $A_k$ at $\sigma'$. Therefore from Definitions 4.7 and 4.9, we see that for every $\sigma' \in Z_{a_i}(\sigma)$ we have that $\mathcal{T}_g, \sigma' \models Kw_{a_i} A_k$. And from the semantics of '$K_{a_i}$' operator (see Definition 4.11), we conclude that $\mathcal{T}_g, \sigma \models K_{a_i} Kw_{a_i} A_k$.

**'$\Longleftarrow$'-direction** Suppose $\mathcal{T}_g, \sigma \models K_{a_i} Kw_{a_i} A_k$ holds. From the semantics of '$K_{a_i}$' operator (see Definition 4.11), this means that for every $\sigma' \in Z_{a_i}(\sigma)$, we have that $\mathcal{T}_g, \sigma' \models Kw_{a_i} A_k$. But $\sigma \in Z_{a_i}(\sigma)$, therefore we conclude that $\mathcal{T}_g, \sigma \models Kw_{a_i} A_k$.

**Case** $i < j < k$**.** We have to prove that:

$$\mathcal{T}_g, \sigma \models Kw_{a_i} A_k \leftrightarrow K_{a_j} Kw_{a_i} A_k$$

**'$\Longrightarrow$'-direction** Suppose $\mathcal{T}_g, \sigma \models Kw_{a_i} A_k$ holds. Then $\sigma$ must be of the form $\sigma = \sigma_1; \ldots; \sigma_m; \ldots$, such that at $\sigma_m$, agent $a_i$ and agent $a_j$ called each other for the first time and exchanged all the secrets they knew at the call (given the constraints of the line topology network, agent $a_i$ cannot learn secret $A_k$, other than through some call with agent $a_j$, and this fact is known by all the agents since the topology of the network is commonly known). Furthermore, from Definitions 4.7 and 4.9, for every $\sigma' \in Z_{a_j}(\sigma)$, we have that $\sigma' = \sigma'_1; \ldots; \sigma'_m; \ldots$ and $\sigma_m \equiv_{a_j} \sigma'_m$, and since agent $a_i$ knows $A_j$ at $\sigma_m$, then from Definition 4.7 we see that agent $a_i$ also knows secret $A_j$ at $\sigma'_m$, for every $\sigma' \in Z_{a_j}(\sigma)$. And from the semantics of '$K_{a_j}$' operator (see Definition 4.11), we conclude that $\mathcal{T}_g, \sigma \models K_{a_j} Kw_{a_i} A_k$.

**'$\Longleftarrow$'-direction** Suppose $\mathcal{T}_g, \sigma \models K_{a_j} Kw_{a_i} A_k$ holds. From the semantics of '$K_{a_j}$' operator (see Definition 4.11), this means that for every $\sigma' \in Z_{a_j}(\sigma)$, we have that $\mathcal{T}_g, \sigma' \models Kw_{a_i} A_k$. But $\sigma \in Z_{a_j}(\sigma)$, therefore we conclude that $\mathcal{T}_g, \sigma \models Kw_{a_i} A_k$.

□

*Proof of Property* (6.15). Given that $a_j \in \overset{\Rightarrow 1}{a_i}$ and given that $i \leq j \leq k$, we distinguish the following cases.

**Case** $i = j = k$**.** Here we have to prove that:

$$\mathcal{T}_g, \sigma \models \neg Kw_{a_i} A_i \leftrightarrow K_{a_i} \neg Kw_{a_i} A_i$$

**'$\Longrightarrow$'-direction** Suppose $\mathcal{T}_g, \sigma \models \neg Kw_{a_i} A_i$ holds. From Definition 4.7, if agent $a_i$ considers $\sigma$ and $\sigma'$ to be equivalent, then agent $a_i$ also knows the same secrets at both $\sigma$ and $\sigma'$; so if agent $a_i$ does not know secret $A_i$ at $\sigma$, then it also does not know secret $A_i$ at $\sigma'$. Therefore from Definitions 4.7 and 4.9, we see that for every $\sigma' \in Z_{a_i}(\sigma)$ we have that $\mathcal{T}_g, \sigma' \models \neg Kw_{a_i} A_i$. And from the semantics of '$K_{a_i}$' operator, we have that $\mathcal{T}_g, \sigma \models K_{a_i} \neg Kw_{a_i} A_i$.

**'$\Longleftarrow$'-direction** Suppose $\mathcal{T}_g, \sigma \models K_{a_i} \neg Kw_{a_i} A_i$ holds. That is, for every $\sigma' \in Z_{a_i}(\sigma)$, we have that $\mathcal{T}_g, \sigma' \models \neg Kw_{a_i} A_i$. But $\sigma \in Z_{a_i}(\sigma)$, therefore we conclude that $\mathcal{T}_g, \sigma \models \neg Kw_{a_i} A_i$.

**Case $i < j = k$.** We have to prove that:

$$\mathcal{T}_g, \sigma \models \neg Kw_{a_i} A_j \leftrightarrow K_{a_j} \neg Kw_{a_i} A_j$$

**'$\Longrightarrow$'-direction** Suppose $\mathcal{T}_g, \sigma \models \neg Kw_{a_i} A_j$ holds. Then $\sigma$ must be of the form $\sigma = \sigma_1; \ldots; \sigma_m; \ldots$, such that there is no such call $\sigma_m$, $1 \leq i$, in which agent $a_i$ and agent $a_j$ called each other to exchange each other's secrets (given the constraints of the line topology network, agent $a_i$ cannot learn secret $A_j$, other than through a call with agent $a_j$, and this fact is known by all the agents since the topology of the network is commonly known). Furthermore, from Definitions 4.7 and 4.9, for every $\sigma' \in Z_{a_j}(\sigma)$, we have that $\sigma' = \sigma'_1; \ldots; \sigma'_m; \ldots$ such that there is no such call $\sigma'_m$, $1 \leq i$, in which agent $a_i$ and agent $a_j$ called each other to exchange each other's secrets. That is, from Definition 4.7, for any execution sequence $\sigma'$ that is equivalent to $\sigma$, for agent $a_j$, there cannot be a call $\sigma'_m$ between agent $a_i$ and $a_j$. In other words, agent $a_j$ does not consider it possible that agent $a_i$ knows secret $A_j$. Therefore, at every $\sigma' \in Z_{a_j}(\sigma)$, agent $a_i$ remains ignorant of secret $A_j$. So from the semantics of '$K_{a_j}$' operator (see Definition 4.11), we conclude that $\mathcal{T}_g, \sigma \models K_{a_j} \neg Kw_{a_i} A_j$.

**'$\Longleftarrow$'-direction** Suppose $\mathcal{T}_g, \sigma \models K_{a_j} \neg Kw_{a_i} A_j$ holds. From the semantics of '$K_{a_j}$' operator (see Definition 4.11), this means that for every $\sigma' \in Z_{a_j}(\sigma)$, we have that $\mathcal{T}_g, \sigma' \models \neg Kw_{a_i} A_j$. But $\sigma \in Z_{a_j}(\sigma)$, therefore we conclude that $\mathcal{T}_g, \sigma \models \neg Kw_{a_i} A_j$.

**Case $i = j < k$.** We have to prove that:

$$\mathcal{T}_g, \sigma \models \neg Kw_{a_i} A_k \leftrightarrow K_{a_i} \neg Kw_{a_i} A_k$$

**'$\Longrightarrow$'-direction** Suppose $\mathcal{T}_g, \sigma \models \neg Kw_{a_i} A_k$ holds. From Definition 4.7, if agent $a_i$ considers any execution sequences $\sigma$ and $\sigma'$ to be equivalent, then agent $a_i$ also knows the same secrets at both $\sigma$ and $\sigma'$; so if agent $a_i$ does not know secret $A_k$ at $\sigma$, then it also does not know secret $A_k$ at $\sigma'$. Therefore from Definitions 4.7 and 4.9, we see that for every $\sigma' \in Z_{a_i}(\sigma)$ we have that

$\mathcal{T}_g, \sigma' \models \neg Kw_{a_i} A_k$. And from the semantics of '$K_{a_i}$' operator (see Definition 4.11), we conclude that $\mathcal{T}_g, \sigma \models K_{a_i} \neg Kw_{a_i} A_k$.

**'$\Longleftarrow$'-direction** Suppose $\mathcal{T}_g, \sigma \models K_{a_i} \neg Kw_{a_i} A_k$ holds. From the semantics of '$K_{a_i}$' operator (see Definition 4.11), this means that for every $\sigma' \in Z_{a_i}(\sigma)$, we have that $\mathcal{T}_g, \sigma' \models \neg Kw_{a_i} A_k$. But $\sigma \in Z_{a_i}(\sigma)$, therefore we conclude that $\mathcal{T}_g, \sigma \models \neg Kw_{a_i} A_k$.

**Case $i < j < k$.** We have to prove that:

$$\mathcal{T}_g, \sigma \models \neg Kw_{a_i} A_k \leftrightarrow K_{a_j} \neg Kw_{a_i} A_k$$

**'$\Longrightarrow$'-direction** Suppose $\mathcal{T}_g, \sigma \models \neg Kw_{a_i} A_k$ holds. Then $\sigma$ must be of the form $\sigma = \sigma_1; \ldots; \sigma_m; \ldots$, such that there is no such call $\sigma_m$, $1 \leq i$, in which agent $a_i$ and agent $a_j$ called each other and at which agent $a_i$ learnt secret $A_k$ (given the constraints of the line topology network, agent $a_i$ cannot learn secret $A_k$, other than through some call with agent $a_j$, and this fact is known by all the agents since the topology of the network is commonly known). Furthermore, from Definitions 4.7 and 4.9, for every $\sigma' \in Z_{a_j}(\sigma)$, we have that $\sigma' = \sigma'_1; \ldots; \sigma'_m; \ldots$ such that there is no such call $\sigma'_m$, $1 \leq i$, in which agent $a_i$ and agent $a_j$ called each other and at which agent $a_i$ learnt secret $A_k$. That is, from Definition 4.7, for any execution sequence $\sigma'$ that is equivalent to $\sigma$, for agent $a_j$, there cannot be a call $\sigma'_m$ between agent $a_i$ and $a_j$ at which agent $a_i$ knows secret $A_k$. In other words, agent $a_j$ does not consider it possible that agent $a_i$ knows secret $A_k$. Therefore, at every $\sigma' \in Z_{a_j}(\sigma)$, agent $a_i$ remains ignorant of secret $A_k$. So from the semantics of '$K_{a_j}$' operator (see Definition 4.11), we conclude that $\mathcal{T}_g, \sigma \models K_{a_j} \neg Kw_{a_i} A_k$.

**'$\Longleftarrow$'-direction** Suppose $\mathcal{T}_g, \sigma \models K_{a_j} \neg Kw_{a_i} A_k$ holds. From the semantics of '$K_{a_j}$' operator (see Definition 4.11), this means that for every $\sigma' \in Z_{a_j}(\sigma)$, we have that $\mathcal{T}_g, \sigma' \models \neg Kw_{a_i} A_k$. But $\sigma \in Z_{a_j}(\sigma)$, therefore we conclude that $\mathcal{T}_g, \sigma \models \neg Kw_{a_i} A_k$.

□

*The proofs for Properties (6.12') - (6.15') of Theorem 6.32 are, respectively, analogous to the proofs for Properties* (6.12) - (6.15).

**Corollary 6.35.** *Given a set of n agents* $\mathsf{Ag} = \{a_1, \ldots, a_n\}$ *on a line topology network, let* $\mathsf{P} = \{A_1, \ldots, A_n\}$ *be the set containing the respective secret of each of the agents. Let* $Kw_{a_i}\mathsf{P}$ *denote* $\bigwedge_{1 \leq j \leq n} Kw_{a_i} A_j$. *Let* $a_1$ *and* $a_n$ *be the two termini of the line topology network, and let* $\mathcal{T}_g$ *be any gossip tree, and* $\sigma$ *be any execution sequence in* $\mathcal{T}_g$. *Then the following property is true in the scenario:* $\mathcal{T}_g, \sigma \models Kw_{a_1}\mathsf{P} \wedge Kw_{a_n}\mathsf{P} \rightarrow \bigwedge_{1<i<n} Kw_{a_i}\mathsf{P}$.

*Proof.* Suppose $\mathcal{T}_g, \sigma \models Kw_{a_1}\mathsf{P}$ and $\mathcal{T}_g, \sigma \models Kw_{a_n}\mathsf{P}$. Let $a_i$ be an arbitrary agent such that $1 < i < n$. Now, from propositional logic, we have that $\mathcal{T}_g, \sigma \models Kw_{a_1}\mathsf{P} \rightarrow Kw_{a_1} A_n$.

So from Property (6.12) (Theorem 6.32) we conclude that:

$$\mathcal{T}_g, \sigma \models \bigwedge_{i \leq k \leq n} K_{a_i} A_k \tag{6.20}$$

Likewise, from propositional logic, we have that $\mathcal{T}_g, \sigma \models Kw_{a_n}\mathsf{P} \rightarrow Kw_{a_n}A_1$. So from Property (6.12') (Theorem 6.32) we conclude that:

$$\mathcal{T}_g, \sigma \models \bigwedge_{1 \leq k \leq i} K_{a_i} A_k \tag{6.21}$$

Therefore given that $\mathcal{T}_g, \sigma \models Kw_{a_1}\mathsf{P}$ and $\mathcal{T}_g, \sigma \models Kw_{a_n}\mathsf{P}$, we conclude, from Properties (6.20) and (6.21) (and from $i$ being arbitrary) that:

$$\mathcal{T}_g, \sigma \models \bigwedge_{1 < i < n} Kw_{a_i}\mathsf{P}$$

□

### 6.3.2 Properties of Epistemic Gossip Protocols on a Line Topology Network

We now establish here results about the behaviour of our epistemic gossip protocols on a line topology network.

**Proposition 6.36.** *Learn New Secrets is not successful for a gossip scenario where the network of agents is a line topology network and the size n of the set of agents is greater than two.*

*Proof.* From the line topology network of agents shown in Figure 6.3, consider the following execution sequence $\sigma$, which is in the extension of Learn New Secrets protocol for any line topology network of agents where the size of the set of agents is greater than two:

$$\sigma \quad = \quad a_1a_2; a_2a_3; \ldots; a_{k-1}a_k; \ldots; a_{n-1}a_n \quad \text{(for all } 3 < k < n\text{)}$$

We see that in each call $a_ia_j$ in $\sigma$, agent $a_i$ calls agent $a_j \in \overset{\Rightarrow 1}{a_i}$. The justification for these calls under Learn New Secrets comes from Property (6.15) of Theorem 6.32. Now, after the last call of $\sigma$, agent $a_{n-1}$ and agent $a_n$ know all the secrets in the scenario, but this is not the case for the other agents. Moreover, under Learn New Secrets, no agent can make any more calls after the last call $a_{n-1}a_n$, since each agent has now learnt the secret of all its neighbours. □

**Proposition 6.37.** *Let agents $a_1, \ldots, a_n$ be connected on a line topology network. Then Known Information Growth de Re is successful on the line topology network.*

*Proof.* Let $\mathcal{T}_g$ be the gossip tree for Known Information Growth de Re. Now take an arbitrary execution sequence $\sigma$ in $\mathcal{T}_g$. Recall that the epistemic calling condition for

agent $a_i$ to call agent $a_j$ at $\sigma$ is:

$$\mathcal{T}_g, \sigma \models K_{a_i} \bigvee_{a_k \in \mathsf{Ag}} K_{a_i}(Kw_{a_i} A_k \nabla Kw_{a_j} A_k) \tag{6.22}$$

First of all, note that during every call, some agent will learn some secret (see Proposition 6.9), so $\sigma$ is terminating. From Definition 4.20, we proceed by contraposition, that is, we will show that if there is some agent $a_i$ and some secret $A_m$ such that $\mathcal{T}_g, \sigma \models \neg Kw_{a_i} A_m$, then the execution sequence $\sigma$ is not terminal.

Let $\sigma = \sigma_1; \ldots; \sigma_p$, and take an agent $a_i$ who does not know a certain secret $A_m$ after some call $\sigma_p$. Without loss of generality, let us assume $i < m$ on the line topology network. Note that $i < ml(\sigma, p, m) \leq m$, since $a_i$ does not know $A_m$ and $i < m$. We know from Property (6.15) that $\mathcal{T}_g, \sigma \models K_{a_{ml(\sigma,p,m)}} \neg Kw_{a_{ml(\sigma,p,m)-1}} A_m$. Since it also holds that $\mathcal{T}_g, \sigma \models Kw_{a_{ml(\sigma,p,m)}} A_m$, the epistemic calling condition for $a_{ml(\sigma,p,m)}$ to call $a_{ml(\sigma,p,m)-1}$ is true. Moreover $a_{ml(\sigma,p,m)}$ and $a_{ml(\sigma,p,m)-1}$ are neighbours on the network graph. We therefore conclude that $\sigma$ is not terminal. □

**Proposition 6.38.** *On a line topology network, and for any gossip tree $\mathcal{T}_g$=$\langle H, R, F, \{Z_{a_m \in \mathsf{Ag}}\}\rangle$, and for any execution sequence $\sigma$ in $\mathcal{T}_g$, the calling conditions for Known Information Growth de Re and for Known Information Growth de Dicto are equivalent.*

*Proof.* Recall that given any epistemic gossip protocol Π, the calling condition for any agent $a_i$ to call another agent $a_j$ is that:

- Both agent $a_i$ and agent $a_j$ are neighbours on the network graph, and
- The epistemic calling condition given by Π for agent $a_i$ to call agent $a_j$ must hold at the considered situation.

But irrespective of the protocol, agent $a_i$ and agent $a_j$ must be neighbours on the network graph for there to be any possibility of a call between both agents. So let us assume that agent $a_i$ and agent $a_j$ are any two agents that are neighbours on a line topology network. Furthermore, assume without loss of generality that the network is as shown in Figure 6.3, and that $j = i + 1$.

Now, recall that the epistemic calling condition for Known Information Growth de Dicto at $\sigma$ is:

$$\mathcal{T}_g, \sigma \models K_{a_i} \bigvee_{a_k \in \mathsf{Ag}} (Kw_{a_i} A_k \nabla Kw_{a_j} A_k) \tag{6.23}$$

So the claim we now want to prove is:

$$\mathcal{T}_g, \sigma \models K_{a_i} \bigvee_{a_k \in \mathsf{Ag}} K_{a_i}(Kw_{a_i} A_k \nabla Kw_{a_j} A_k) \leftrightarrow K_{a_i}(\bigvee_{a_k \in \mathsf{Ag}} Kw_{a_i} A_k \nabla Kw_{a_j} A_k) \tag{6.24}$$

**'$\Longrightarrow$' -direction.** Suppose:

$$\mathcal{T}_g, \sigma \models K_{a_i} \bigvee_{a_k \in \mathsf{Ag}} K_{a_i}(Kw_{a_i} A_k \nabla Kw_{a_j} A_k) \tag{6.25}$$

First of all, recall Proposition 4.24 and note that in **S5**, $K_{a_i}(K_{a_i}\alpha \vee K_{a_i}\beta)$ is equivalent to $(K_{a_i}\alpha \vee K_{a_i}\beta)$ (for '$\Rightarrow$', use veridicality, for '$\Leftarrow$', observe that $K_{a_i}\alpha$ implies $K_{a_i}K_{a_i}\alpha$ which in turn implies $K_{a_i}(K_{a_i}\alpha \vee K_{a_i}\beta)$). Likewise, $K_{a_i}\beta$ implies $K_{a_i}(K_{a_i}\alpha \vee K_{a_i}\beta)$, from which the result follows (†).

Now let $\mathsf{Ag} = \{a_1, \ldots, a_n\}$. Let $\alpha_r = (Kw_{a_i}A_r \nabla Kw_{a_j}A_r)$. Therefore we have:

$$
\begin{array}{lll}
\mathcal{T}_g, \sigma & \models K_{a_i} \bigvee\limits_{a_k \in \mathsf{Ag}} K_{a_i}(Kw_{a_i}A_k \nabla Kw_{a_j}A_k) & \\
\iff & \mathcal{T}_g, \sigma \models K_{a_i}(K_{a_i}\alpha_1 \vee \cdots \vee K_{a_i}\alpha_n) & \\
\iff & \mathcal{T}_g, \sigma \models (K_{a_i}\alpha_1 \vee \cdots \vee K_{a_i}\alpha_n) & \text{(by (†))} \\
\Longrightarrow & \mathcal{T}_g, \sigma \models K_{a_i}\alpha_j \quad \text{for some } 1 \leq j \leq n & \\
\Longrightarrow & \mathcal{T}_g, \sigma' \models \alpha_j \text{ for every } \sigma' \in Z_{a_i}(\sigma) & \text{(from Definition 4.11)} \\
\Longrightarrow & \mathcal{T}_g, \sigma' \models (\alpha_1 \vee \cdots \vee \alpha_j \vee \cdots \vee \alpha_n) \quad \text{for every } \sigma' \in Z_{a_i}(\sigma) & \\
 & & \text{(by propositional logic)} \\
\Longrightarrow & \mathcal{T}_g, \sigma \models K_{a_i}(\alpha_1 \vee \cdots \vee \alpha_n) & \text{(from Definition 4.11)} \\
\Longrightarrow & \mathcal{T}_g, \sigma \models K_{a_i}( \bigvee\limits_{a_k \in \mathsf{Ag}} Kw_{a_i}A_k \nabla Kw_{a_j}A_k) &
\end{array}
$$

**'$\Longleftarrow$' -direction.** Recall that we have a line topology network. Suppose that agent $a_i$ and agent $a_j$ are neighbours, say agent $a_j$ is at the immediate right of agent $a_i$ (the other case is similar)[†]. Suppose $\mathcal{T}_g, \sigma \models K_{a_i} \bigvee_{a_k \in \mathsf{Ag}}(Kw_{a_i}A_k \nabla Kw_{a_j}A_k)$. This means that:

$$
\begin{array}{lll}
\mathcal{T}_g, \sigma \models K_{a_i}( & (Kw_{a_i}A_1 \wedge \neg Kw_{a_j}A_1) \vee (\neg Kw_{a_i}A_1 \wedge Kw_{a_j}A_1) \vee & \\
 & (Kw_{a_i}A_2 \wedge \neg Kw_{a_j}A_2) \vee (\neg Kw_{a_i}A_2 \wedge Kw_{a_j}A_2) \vee & \\
 & \quad \ldots \vee \quad \ldots \vee & \\
 & (Kw_{a_i}A_i \wedge \neg Kw_{a_i}A_i) \vee (\neg Kw_{a_i}A_i \wedge Kw_{a_j}A_i) \vee & \\
 & (Kw_{a_i}A_j \wedge \neg Kw_{a_j}A_j) \vee (\neg Kw_{a_i}A_j \wedge Kw_{a_j}A_j) \vee & \\
 & \quad \ldots \vee \quad \ldots \vee & \\
 & (Kw_{a_i}A_n \wedge \neg Kw_{a_j}A_n) \vee (\neg Kw_{a_i}A_n \wedge Kw_{a_j}A_n) \vee & \\
) & &
\end{array}
\tag{6.26}
$$

[†]Note that since we have a line topology network, if $a_i$ and $a_j$ are not neighbours then the calling condition for $a_i$ to call $a_j$ (or vice versa) fails immediately for any protocol.

Because of (6.12'), and since agent $a_i$ knows that $Kw_{a_i}A_i$, this gives:

$$\begin{array}{lrcl} \mathcal{T}_g,\sigma \models K_{a_i}( & (Kw_{a_i}A_1 \wedge \neg Kw_{a_j}A_1) & \vee & \bot & \vee \\ & (Kw_{a_i}A_2 \wedge \neg Kw_{a_j}A_2) & \vee & \bot & \vee \\ & \dots & \vee & \dots & \vee \\ & (Kw_{a_i}A_i \wedge \neg Kw_{a_j}A_i) & \vee & \bot & \vee \\ & (Kw_{a_i}A_j \wedge \neg Kw_{a_j}A_j) & \vee & (\neg Kw_{a_i}A_j \wedge Kw_{a_j}A_j) & \vee \\ & \dots & \vee & \dots & \vee \\ & (Kw_{a_i}A_n \wedge \neg Kw_{a_j}A_n) & \vee & (\neg Kw_{a_i}A_n \wedge Kw_{a_j}A_n) & \vee \\ ) & & & & \end{array} \tag{6.27}$$

Moreover, since $a_i$ knows that $Kw_{a_j}A_j$, and because of (6.12), we obtain:

$$\begin{array}{lrcl} \mathcal{T}_g,\sigma \models K_{a_i}( & (Kw_{a_i}A_1 \wedge \neg Kw_{a_j}A_1) & \vee & \bot & \vee \\ & (Kw_{a_i}A_2 \wedge \neg Kw_{a_j}A_2) & \vee & \bot & \vee \\ & \dots & \vee & \dots & \vee \\ & (Kw_{a_i}A_i \wedge \neg Kw_{a_j}A_i) & \vee & \bot & \vee \\ & \bot & \vee & (\neg Kw_{a_i}A_j \wedge Kw_{a_j}A_j) & \vee \\ & \dots & \vee & \dots & \vee \\ & \bot & \vee & (\neg Kw_{a_i}A_n \wedge Kw_{a_j}A_n) & \vee \\ ) & & & & \end{array} \tag{6.28}$$

Writing $\alpha_m$ for $Kw_{a_i}A_m \wedge \neg Kw_{a_j}A_m$ and $\beta_p$ for $\neg Kw_{a_i}A_p \wedge Kw_{a_j}A_p$, we can simplify (6.28) to:

$$\mathcal{T}_g,\sigma \models K_{a_i}(\bigvee_{1\leq m\leq i} \alpha_m \vee \bigvee_{j\leq p\leq n} \beta_p) \tag{6.29}$$

For every $\alpha_m$, we have that $\mathcal{T}_g,\sigma \models \alpha_m \rightarrow K_{a_i}\bigvee_{a_k\in \mathsf{Ag}}(Kw_{a_i}A_k \nabla Kw_{a_j}A_k)$, and also $\mathcal{T}_g,\sigma \models \alpha_m \leftrightarrow K_{a_i}\alpha_m$ ('$\Rightarrow$' follows from (6.14') and (6.15'); '$\Leftarrow$' follows from veridicality). This implies that if one of the $\alpha_m$ formulas holds, we are done. Moreover, if none of the $\alpha_m$ formulas is true, agent $a_i$ knows this, and hence we obtain that $\mathcal{T}_g,\sigma \models K_{a_i}\bigvee_{j\leq p\leq n}\beta_p$. Let $q$ be the highest index for which $\mathcal{T}_g,\sigma \models Kw_{a_i}A_q$ holds. Then we have that (6.29) is equivalent to $\mathcal{T}_g,\sigma \models K_{a_i}(\bigvee_{q<p\leq n}\beta_p)$. Since on a line it holds that $\mathcal{T}_g,\sigma \models K_{a_i}(Kw_{a_j}A_x \rightarrow Kw_{a_j}A_{x-1})$ if $i<j<x$ (from (6.13), which is common knowledge among all the agents, since, according to Theorem 6.32, (6.13) holds at all possible execution sequences in $\mathcal{T}_g$); we then obtain that $\mathcal{T}_g,\sigma \models K_{a_i}\beta_{q+1}$ holds, that is, $\mathcal{T}_g,\sigma \models K_{a_i}(\neg Kw_{a_i}A_{q+1} \wedge Kw_{a_j}A_{q+1})$.

This concludes the proof. □

**Corollary 6.39.** *For a gossip scenario on a line topology network, the extension of Known Information Growth de Re is equal to the extension of Known Information Growth de Dicto.*

*Proof.* The proof follows directly from Proposition 6.38. □

**Corollary 6.40.** *Known Information Growth de Dicto is successful in a gossip scenario in which the network of agents is a line topology network.*

*Proof.* From Proposition 6.38, the epistemic calling condition of Known Information Growth de Re is equivalent to the epistemic calling condition of Known Information Growth de Dicto, on a line topology network. Therefore every execution sequence of Known Information Growth de Re is also an execution sequence of Known Information Growth de Dicto. Therefore, from Proposition 6.37, since Known Information Growth de Re is successful on a line topology network, we conclude that Known Information Growth de Dicto is also successful on a line topology network. □

**Proposition 6.41.** *Given a gossip scenario with $n$ agents on a line topology network, the minimum length of a successful execution sequence of Known Information Growth de Dicto and Known Information Growth de Re is $2n-3$.*

*Proof.* We begin by constructing an execution sequence, and then we show that the constructed execution sequence is an execution sequence of Known Information Growth de Re and Known Information Growth de Dicto. Finally we show, from Literature, that the constructed execution sequence is of the shortest successful execution sequence length.

Let the network of agents be as shown in Figure 6.3. Consider the following execution sequence $\sigma$ given as follows.

$$\sigma = \sigma_1; \sigma_2$$

where

$$\begin{array}{lcll} \sigma_1 & = & a_1a_2; \ldots; a_ka_{k+1}; \ldots; a_{n-1}a_n & \text{(for all } 2 \leq k < n-1\text{)} \\ \sigma_2 & = & a_{n-1}a_{n-2}; \ldots; a_ka_{k-1}; \ldots; a_2a_1 & \text{(for all } n-2 \geq k > 2\text{)} \end{array}$$

After each call $a_{i-1}a_i$ in $\sigma_1$, agent $a_i$ learns the secret of agent $a_{i-1}$, so $K_{a_i}A_{i-1}$ holds at the end of such $a_{i-1}a_i$ call in $\sigma_1$, although agent $a_{i+1}$ does not yet know secret $A_{i-1}$ at this point. So we know from Property (6.15') that $K_{a_i}\neg Kw_{a_{i+1}}A_{i-1}$ holds at the end of the $a_{i-1}a_i$ call in $\sigma_1$, and so the calling condition for agent $a_i$ to call agent $a_{i+1}$ is true for Known Information Growth de Re, at the end of the $a_{i-1}a_i$ call in $\sigma_1$.

Similarly, after each $a_{k+1}a_k$ call in $\sigma_2$, agent $a_k$ learns the secret of agent $a_{k+1}$, so $K_{a_k}A_{k+1}$ holds at the end of such $a_{k+1}a_k$ call in $\sigma_2$, although agent $a_{k-1}$ does not yet know secret $A_{k+1}$ at that point. So we know from Property (6.15) that $K_{a_k}\neg Kw_{a_{k-1}}A_{k+1}$ holds at the end of the $a_{k+1}a_k$ call in $\sigma_2$, and so the calling condition for agent $a_k$ to call agent $a_{k-1}$ is true for Known Information Growth de Re, at the end of the $a_{k+1}a_k$ call in $\sigma_2$.

From the foregoing we see that $\sigma$ is an execution sequence of Known Information Growth de Re. Observe also that the length of $\sigma_1$ is $n-1$, and the length of $\sigma_2$ is $n-2$. So the total length of $\sigma$ is $2n-3$. Observe also that the execution sequence $\sigma$ is successful.

Now, Farley and Proskurowski [23] proved that on a line topology network no successful execution sequence is less than $2n-3$. Therefore, from the foregoing, we conclude that

the minimum length of a successful execution sequence of Known Information Growth de Re on a line topology network is $2n - 3$.

Furthermore, from Corollary 6.39, we conclude also that the minimum length of a successful execution sequence of Known Information Growth de Dicto on a line topology network is $2n - 3$. □

**Proposition 6.42.** *Possible Information Growth de Re is non-terminating in a gossip scenario where the network of agents is a line topology network and the size n of the set of agents is greater than three.*

*Proof.* From the line topology network of agents shown in Figure 6.3, consider the following execution sequence $\sigma$, which is in the extension of Possible Information Growth protocol for any line topology network of agents where the size of the set of agents is greater than three:

$$\sigma \quad = \quad a_1a_2; a_3a_4; a_1a_2; a_4a_3; a_1a_2; a_4a_3; \ldots$$

The first two calls of $\sigma$ are justified by Property (6.15) of Theorem 6.32: in the first call in $\sigma$ agent $a_1$ *knows*, and therefore considers it *possible* (from the possible world semantics of knowledge) that it will learn the secret of agent $a_2$ in the first call; and similarly, in the second call, agent $a_3$ knows that it will learn the secret of agent $a_4$ in the second call with agent $a_4$. In the third call, agent $a_1$ considers it possible that agent $a_2$ learnt some secret in the second call (in a possible $a_2a_3$ call which is justified by Property (6.15) of Theorem 6.32) and so agent $a_1$ calls agent $a_2$ in the third call, which proves redundant. Likewise in the fourth call, agent $a_4$ considers it possible that agent $a_3$ learnt some secret in the third call (in a possible call with agent $a_2$) and therefore agent $a_4$ calls agent $a_3$ in the fourth call, which also proves redundant. Once again agent $a_1$ considers it possible that agent $a_2$ learnt some new secret in the preceding call, and thus for the execution sequence $\sigma$, the loop $a_1a_2; a_4a_3$ goes on infinitely. □

**Proposition 6.43.** *Possible Information Growth de Dicto is non-terminating in a gossip scenario where the network of agents is a line topology network and the size n of the set of agents is greater than three.*

*Proof.* The proof follows from the proof of Proposition 6.42, and from the proof of Proposition 3.27 (and Observation 4.26) which shows that the extension of Protocol 4 is equal to the extension of Protocol 5. □

## 6.4 Tree Topology Network

Let us now consider a tree topology network of agents in a gossip scenario, and study the properties of Learn New Secrets, Known Information Growth de Dicto, Known Information Growth de Re, Possible Information Growth de Dicto and Possible Information Growth de Re. See Figures 6.5 and 6.6 for examples of a tree topology network.

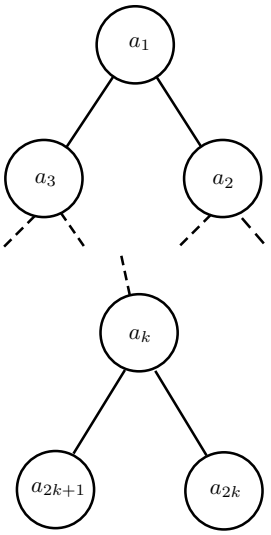



FIGURE 6.5: **Binary Tree Topology Network.**

**Proposition 6.44.** *Known Information Growth de Dicto and Known Information Growth de Re are successful in a gossip scenario in which the network of agents is a tree topology network.*

*Proof.* We begin by proving the proposition for Known Information Growth de Re. Let $\mathcal{T}_g$ be the gossip tree for Known Information Growth de Re on a tree topology network. Now take an arbitrary execution sequence $\sigma$ in $\mathcal{T}_g$. Recall that the epistemic calling condition for agent $a_i$ to call agent $a_j$ at $\sigma$ is:

$$\mathcal{T}_g, \sigma \models K_{a_i} \bigvee_{a_k \in \mathsf{Ag}} K_{a_i}(Kw_{a_i}A_k \nabla Kw_{a_j}A_k)$$

Take an arbitrary execution sequence $\sigma$ of any of the two protocols given. First of all, note that during every call, some agent will learn some secret, (see Proposition 6.9), so $\sigma$ is terminating. From Definition 4.20, we proceed by contraposition, that is, we will show that if there is some agent $a_i$ and some secret $A_m$ such that $\mathcal{T}_g, \sigma \models \neg Kw_{a_i}A_m$, then the execution sequence $\sigma$ is not terminal.

Let $\sigma = \sigma_1; \ldots; \sigma_p$, and take an agent $a_i$ who does not know a certain secret $A_m$ after some call $\sigma_p$. Now, consider the path from agent $a_i$ to agent $a_m$ on the tree. This path is a line topology network, and agent $a_i$ and agent $a_m$ are the termini of such line network. Without loss of generality, let us assume $i < m$ on the line topology network. Note that $i < ml(\sigma, p, m) \leq m$, since $a_i$ does not know $A_m$ and $i < m$. We know from Property (6.15) that $\mathcal{T}_g, \sigma \models K_{a_{ml(\sigma,p,m)}} \neg Kw_{a_{ml(\sigma,p,m)-1}} A_m$. Since it also holds that $\mathcal{T}_g, \sigma \models Kw_{a_{ml(\sigma,p,m)}} A_m$, the calling condition for $a_{ml(\sigma,p,m)}$ to call $a_{ml(\sigma,p,m)-1}$ is true. Moreover $a_{ml(\sigma,p,m)}$ and $a_{ml(\sigma,p,m)-1}$ are neighbours on the network graph. We therefore conclude that $\sigma$ is not terminal. So we also conclude that Known Information Growth de Re is successful on a tree topology network.

And based on the foregoing argument, and on Proposition 6.38, we conclude also that Known Information Growth de Dicto is successful on a tree topology network. □

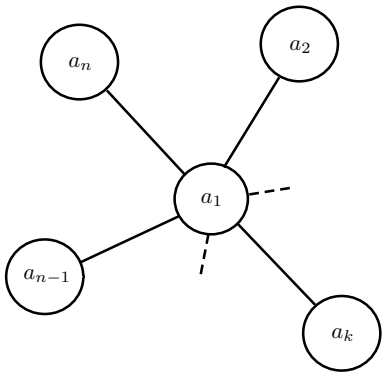



FIGURE 6.6: **Star Topology Network.**

**Corollary 6.45.** *Known Information Growth de Re and Known Information Growth de Dicto are successful in a gossip scenario in which the network of agents is either a binary tree topology network or a star topology network.*

*Proof.* This follows from the proof of Proposition 6.44, since both the binary tree topology network and the star topology network are instances of a tree topology network. □

Both Known Information Growth de Dicto and Known Information Growth de Re are successful on an arbitrary connected network in which there are no cycles. This fact follows from Proposition 6.44, since such an arbitrary connected network is a tree topology network.

**Proposition 6.46.** *Learn New Secrets protocol is not successful for a gossip scenario where the network of agents is not a complete topology network and the size n of the set of agents is greater than two.*

*Proof.* Consider the following protocol.

> *For every $a_i \in \mathsf{Ag}$, let $a_i$ call all its neighbours whom it does not yet know their secret.*

Every sequence of the given protocol is an execution sequence of Learn New Secrets protocol, because, for each call, the protocol directly requires the epistemic calling condition of Learn New Secrets. But since the network is not complete topology network, there is some pair of nodes $a_i, a_j$ on the network graph such that there is no edge between $a_i$ and $a_j$. Now consider an execution sequence $\sigma$ of the given protocol that begins by selecting such agent $a_i$ to call all its neighbours in turn, and then the execution sequence $\sigma$ continues by selecting each of the other agents in turn to call their own neighbours of whom they do not yet know their secrets. The execution sequence $\sigma$ is terminal since

after all its calls, no further calls can be made by any agent as they have learnt the secret of all their neighbours. However then, $\sigma$ is unsuccessful since $a_i$ will never learn all the secrets in the scenario, and this is because after the initial calls from agent $a_i$ to all its neighbours, none of $a_i$'s neighbours can repeat a call to $a_i$ after they have learnt the secrets of their other neighbours, since they already know the secret of $a_i$. Since such a sequence is a terminal execution sequence of Learn New Secrets for the given network, we conclude that Learn New Secrets is unsuccessful in such a network. □

**Corollary 6.47.** *Learn New Secrets protocol is not successful for a gossip scenario where the network of agents is a tree topology network and the size $n$ of the set of agents is greater than two.*

*Proof.* The proof follows immediately from Proposition 6.46. □

**Proposition 6.48.** *Possible Information Growth de Re is not successful in a gossip scenario where the network of agents is a tree topology network and the size $n$ of the set of agents is greater than three.*

*Proof.* Consider any agents $a_1$ and $a_m$ such that the path $P$ starting from $a_1$ and ending in $a_m$ is the longest path on the tree. The path $P$ is a line topology network. Let $\mu = |P|$ denote the number of agents along $P$. Consider the following cases:

*Case 1*: Let $\mu > 3$. Then, from Proposition 6.42, it follows that the agents on such a path are not guaranteed to know all the secrets of the agents along that path.

*Case 2*: Let $\mu = 3$. Then, there is some agent $a_k$ on $P$, and some other agent $a_l$ such that $a_l$ is a child node of $a_k$ in the tree, and $a_l$ is not on $P$. Let the path $P = \langle a_1, a_2, a_3 \rangle$, and let $a_l$ be a child node of $a_2$ (else $\mu > 3$, which is not the case we consider). Therefore we consider the following execution sequence $\sigma$, which is in the extension of Possible Information Growth protocol for any such tree topology network of agents:

$$\sigma \quad = \quad a_1a_2; a_3a_2; a_1a_2; a_3a_2; a_1a_2; a_3a_2; \ldots$$

The first two calls of $\sigma$ are justified respectively by Properties (6.15') and (6.15) of Theorem 6.32: in the first call in $\sigma$ agent $a_1$ *knows*, and therefore considers it *possible* (from the possible world semantics of knowledge) that it will learn secret $A_2$ in the first call; and similarly, in the second call, agent $a_3$ knows that it will learn secret $A_2$ in a call with agent $a_2$. In the third call, agent $a_1$ considers it possible that agent $a_2$ learnt some secret in the second call (in a possible call with agent $a_3$), and so agent $a_1$ calls agent $a_2$ in the third call. Likewise in the fourth call, agent $a_3$ considers it possible that $a_2$ learnt some secret in the third call (in a possible call with agent $a_l$) and therefore $a_3$ calls agent $a_2$ in the fourth call which now proves redundant. Once again agent $a_1$ considers it possible that agent $a_2$ learnt some new secret in the preceding call (again, in a possible

call with agent $a_l$), so agent $a_1$ calls $a_2$ again, which also proves redundant. Thus for the execution sequence $\sigma$, the loop $a_1a_2; a_3a_2; a_1a_2; a_3a_2; \ldots$ goes on infinitely. □

**Proposition 6.49.** *Possible Information Growth de Dicto is non-terminating in a gossip scenario where the network of agents is a tree topology network and the size n of the set of agents is greater than three.*

*Proof.* The proof follows from the proof of Proposition 6.48, and from the proof of Proposition 3.27 (and Observation 4.26) which shows that the extension of Protocol 4 is equal to the extension of Protocol 5. □

### 6.4.1 Synthesising a Maximum Length Successful Execution Sequence on a Tree Topology Network

Given a tree topology network, let us consider the length of the longest successful execution sequence of calls under our epistemic gossip protocols.

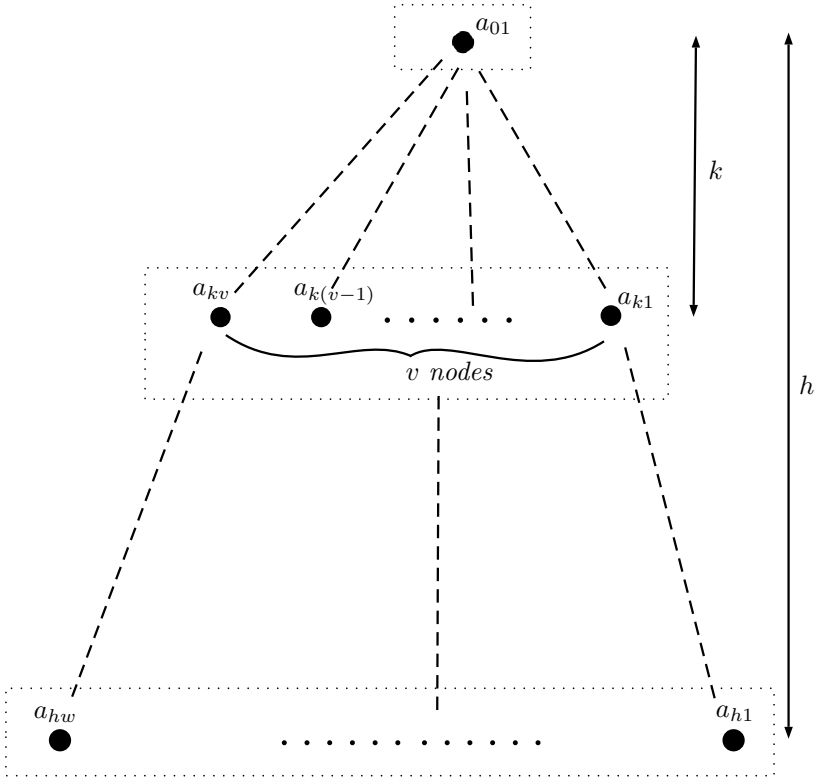



FIGURE 6.7: **Tree topology network setting for Procedure 6.3.**

Let the nodes on a tree topology network be as shown in Figure 6.7. Note the following about the arrangement of nodes in Figure 6.7.

- We group the nodes in layers as follows: the nodes at a distance $y$ from the root node belong to layer $y$.

- Nodes in the same layer are labelled, right-to-left beginning from one, in a unit stepwise increasing index which represents the horizontal position of the nodes within the layer. For any node $a_{ij}$ in the tree of Figure 6.7, $i$ is the distance of the node from the root node and $j$ is the horizontal position of the node within layer $i$. For example, the root node is labelled $a_{01}$, and suppose there are $v$ nodes in layer $k$, then the rightmost node is labelled $a_{k1}$ and the leftmost node is labelled $a_{kv}$.
- Finally, note that $h$ is the height of the tree topology network, that is the maximum distance of any node from the root of the tree.

**Procedure 6.2** Bubble and Breadth-First procedure.

**function** BUBBLE($a_{yx}$)
    if $a_{yx} \neq a_{01}$, then
        Let $a_{yx}$ call its parent node $a_{yx.parent}$
        BUBBLE($a_{yx.parent}$)
    end if
    return
**end function**

**function** BFP($a_{yx}, a_{st}$)
    BUBBLE($a_{yx}$);
    Let $g$ be the greatest node index that is less than $yx$.
    for $a_{ij} = a_{01}$ to $a_g$ and $a_{ij} \neq a_{yx}$
        Except for the reverse of any call made during BUBBLE($a_{yx}$), let $a_{ij}$ call all
            its child nodes in order of increasing node index.
    end for
    Let $a_{yx}$ call all its child nodes whose index is lower than that of $a_{st}$, in order of
            increasing node index.
**end function**

**Procedure 6.3** Longest sequence procedure for a tree topology network.

Given a tree topology network labelled as shown in Figure 6.7.
Let the agents be at the initial state.
 for $i = 0$ to $h - 1$
    Let the number of nodes in layer $i$ be $w$
     for $a_{kl} = a_{i1}$ to $a_{iw}$
         for each child node $a_{ch}$ of $a_{kl}$, in order of increasing node index
             Let $a_{kl}$ call $a_{ch}$
             BFP($a_{kl}, a_{ch}$)
         end for
    end for
end for

### Description of Procedure 6.3

Procedure 6.3 describes an execution sequence of calls among agents on a tree topology network as shown in Figure 6.7. Execution begins with the root node calling its child

nodes in turn, in order of increasing node index. Each subsequent node gets to call its child nodes too. After each call $a_{kl}a_{ch}$ with a child node, agent $a_{kl}$ begins a subsequence of calls which transmits the secrets learnt from $a_{ch}$ through a relay of parent nodes, up to the root node. This upward relay of the secret of $a_{ch}$ to the root node by $a_{kl}$, is described in the Bubble function (see Procedure 6.2). When the root node learns the secret of $a_{ch}$, this secret is then distributed to all other nodes whose index is less than that of $a_{kl}$, through the BFP function.

Given a scenario with three agents as shown in Figure 6.8, Procedure 6.3 gives rise to the following execution sequence:

$$\sigma_3 = a_{01}a_{11}; a_{01}a_{12}; a_{01}a_{11}$$

In $\sigma_3$, the first call $a_{01}a_{11}$ is directly from Procedure 6.3. The first layer is layer 0, and the rightmost (and only) agent in that layer is $a_{01}$, so we begin by letting $a_{01}$ call its rightmost child node $a_{11}$, as described in Lines 5-7 of Procedure 6.3. After this the BFP function is called with $a_{01}$ as the first parameter, and $a_{11}$ as the second parameter. Now the first line under BFP is a call to Bubble on $a_{01}$, which immediately terminates without any calls since $a_{01}$ is the root node. The 'for' loop in function BFP is also not executed since at $a_{ij} = a_{01}$ we also have that $a_{ij} = a_{yx}$ (since $a_{yx} = a_{01}$). The other values for $a_{ij}$ are not possible at this point: there is no such node whose index is less than $a_{01}$. So we have that the 'for' loop in Lines 11-14 is skipped, and execution resumes at Line 15 of function BFP. But since there is no child node of $a_{01}$ whose index is lower than that of $a_{11}$, this line is also skipped, and we then return to the 'for' loop in Line 6 of Procedure 6.3. At this point, the next child node of $a_{01}$ is $a_{12}$, and as a result of Line 7 we have the second call of $\sigma_3$. Then BFP is again called at Line 8, with $a_{01}$ as the first parameter, and $a_{12}$ as the second parameter. Again, the call to function Bubble (at Line 9 of function BFP) terminates immediately since it starts already at the root node, $a_{01}$. For the same reason as in the foregoing case, the 'for' loop in function BFP is also not executed. So execution resumes at Line 15 of function BFP. At this point $a_{01}$ makes a call to $a_{11}$ and the function exits (there are no other child nodes whose index is lower than that of $a_{11}$). We now return to Line 5 of Procedure 6.3. There are no more nodes in layer $i = 0$, so we return to the 'for' loop in Line 3, where we find that $i = h - 1 = 0$ since $h = 1$. So the loop terminates and the execution of Procedure 6.3 stops.

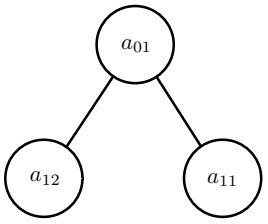



FIGURE 6.8: **Three agents in a tree topology network.**

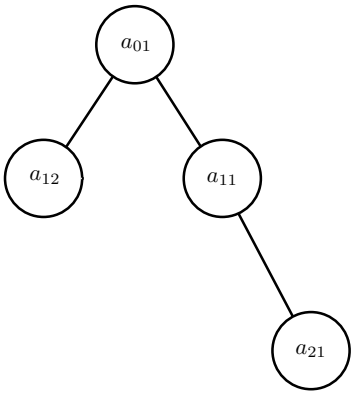



FIGURE 6.9: **Four agents in a tree topology network.**

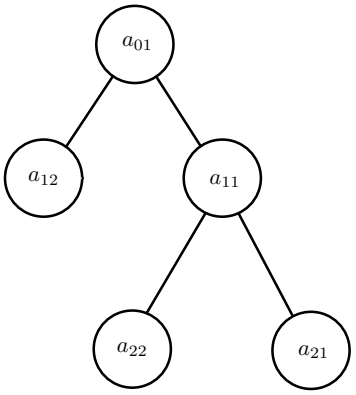



FIGURE 6.10: **Five agents in a tree topology network.**

For a scenario with four agents as shown in Figure 6.9, Procedure 6.3 gives rise to the following execution sequence:

$$\sigma_4 = \sigma_3; a_{11}a_{21}; a_{11}a_{01}; a_{01}a_{12}$$

Notice that the first three calls of the four-agent scenario is the same as those of the three-agent scenario. In this case however, the height $h$ of the tree is 2, so execution of Procedure 6.3 continues where $\sigma_3$ stopped, that is, the loop at Line 3 is continued with $i = 1$. So at Line 7, $a_{11}$ calls its rightmost child node $a_{21}$, yielding the fourth call of $\sigma_4$. Then follows a call to BFP at Line 8, with $a_{11}$ as the first parameter and $a_{21}$ as the second parameter. The first line (Line 9) of function BFP is a call to function Bubble with $a_{11}$ as the parameter. At function Bubble, the fifth call $a_{11}a_{01}$ takes place, and the Bubble function exits ($a_{01}$ is the parent of $a_{11}$, and at the same time the root node). We return to Line 11 of function BFP. At the 'for' loop at Line 11 we have the following: the call $a_{01}a_{11}$ is omitted since its reverse ($a_{11}a_{01}$) was executed in the Bubble function that just exited; but the call $a_{01}a_{12}$ is executed (its reverse was not executed in the Bubble function) - this gives the sixth call of $\sigma_4$; according to the 'for' loop condition at Line 11,

the node with the greatest index that is less than that of $a_{11}$ is $a_{01}$, and since we have treated the cases for all the child nodes of $a_{01}$, the loop stops, and we arrive at Line 15 of function BFP. At this point, we also have that no child node of $a_{11}$ has an index that is lower than that of $a_{21}$, so no call is executed at this point, and the function exits. We return to the Procedure 6.3 at the 'for' loop at Line 5, but since there are no more nodes on layer $i = 1$, this loop exits; the loop on Line 3 also exits since $i$ is now equal to $h - 1$. And here Procedure 6.3 finishes, finally yielding the calls of $\sigma_4$.

Finally, for a scenario with five agents as shown in Figure 6.10, Procedure 6.3 gives rise to the following execution sequence:

$$\sigma_5 = \sigma_4; a_{11}a_{22}; a_{11}a_{01}; a_{01}a_{12}; a_{11}a_{21}$$

The description of how Procedure 6.3 yields $\sigma_5$ is similar to previous examples.

**The length of the execution sequence generated by Procedure 6.3**

Given a scenario with $n$ agents, how many calls has the resulting execution sequence from Procedure 6.3?

Let us for now ignore the call to function BFP on Line 8 of Procedure 6.3, then we see that beginning from layer 0, until layer $h - 1$, each node calls each of all its child nodes exactly once. This yields a total of $(n - 1)$, corresponding to using each of the edges of the tree exactly once.

Now consider the call to function BFP with $a_{kl}$ as the first parameter and $a_{ch}$ as the second parameter, where $a_{kl}$ is the node that just called one of its child nodes $a_{ch}$ in Line 7 of Procedure 6.3. Function BFP begins with a call to function Bubble, with node $a_{kl}$ as parameter, and function Bubble yields a number of calls corresponding to the distance of $a_{kl}$ from the root node, and this distance is given by the value of $k$. Furthermore, in Lines 11-16, we see that beginning from the root node, until node $a_{kl}$, each node calls all its child nodes, except for (a) calls that are the reverse call of one of the calls made during function Bubble, *and* (b) node $a_{kl}$ which calls only the child nodes whose index is less than that of node $a_{ch}$. So considering the edges that were used during the function Bubble, we see that function BFP uses, exactly once, all the edges between all the nodes whose index are less than that of $a_{ch}$. So now, suppose the nodes of the tree are assigned serial natural numbers in order of increasing node index, beginning from number one (for example, the root node will be assigned the number 1, and the leftmost node at layer $h$ will assigned the number $n$). So with this new numbering, let the number assigned to $a_{ch}$ be $m$, then we see that function BFP$(a_{kl}, a_{ch})$ uses, exactly once, all the edges between node number 1 and node number $m - 1$, and this gives a total use of $m - 2$ edges (or calls) for each call of function BFP, where $m$ is the serial natural node number of node $a_{ch}$.

But the serial number of $a_{ch}$ increases by 1 each time function BFP is called, from the order imposed by the 'for' loops in Procedure 6.3. Moreover, from the forgoing analysis,

the first call to function BFP uses $m-2 = 2-2 = 0$ edges (or calls) (since node $a_{11}$ would be assigned the natural number 2, and the first call to function BFP occurs right after $a_{01}$ calls $a_{11}$). Therefore, for each serial number $m$ ranging as follows: $2 \leq m \leq n$, we have that function BFP uses $m-2$ edges (or calls). So the summation of these number of calls over the range of values of $m$ is:

$$\sum_{m=2}^{n}(m-2) = \frac{(n-1)(n-2)}{2}$$

Now recall that, without the call to function BFP on Line 8 of Procedure 6.3, we obtain a total of $n-1$ calls. Therefore the total number of calls in the call sequence generated by Procedure 6.3 is given by:

$$\frac{(n-1)(n-2)}{2} + (n-1) = \frac{n(n-1)}{2}$$

Continuing with our study of Procedure 6.3, consider the following claim.

**Proposition 6.50.** *For a gossip scenario with $n$ agents, the call sequence generated by Procedure 6.3 is an execution sequence of Known Information Growth de Re.*

*Proof.* Recall that the calling condition for Known Information Growth de Re states that for any agent $a_i$ to call another agent $a_j$, there must be a secret $A_k$ such that $a_i$ knows that one of $a_i$ and $a_j$ will learn $A_k$ from the $a_ia_j$ call; moreover, agent $a_i$ and $a_j$ must be neighbours on the network graph. Now let us consider whether this is the case for each of the calls in the call sequence generated by Procedure 6.3. We proceed by examining Procedure 6.3 as follows.

1. Let us for now ignore the call to function BFP on Line 8 of Procedure 6.3, then we see that beginning from layer 0, until layer $h-1$, each node calls each of all its child nodes exactly once. In each of those calls, the node calling its child node knows that it will learn the secret of its child node in that call, since it is the first time they call each other. Hence the calling condition for Known Information Growth de Re is satisfied for each of those calls.

2. Now consider the call to function BFP with $a_{kl}$ as the first parameter and $a_{ch}$ as the second parameter, where $a_{kl}$ is the node that just called one of its child nodes $a_{ch}$ in Line 7 of Procedure 6.3. Obviously $a_{kl}$ learnt the secret of $a_{ch}$ in that call preceding the execution of function BFP. Now, function BFP begins with a call to function Bubble, with node $a_{kl}$ as parameter. Suppose the parent node of $a_{kl}$ is $a_{kl.parent}$, then the path beginning from node $a_{ch}$ and terminating on node $a_{kl.parent}$ is a line topology network comprising of the three agents $a_{ch}, a_{kl}$ and $a_{kl.parent}$. Then from Property 6.15' (or Property 6.15, as the case may be) we see that $a_{kl}$ knows that $a_{kl.parent}$ does not know the secret of $a_{ch}$, and therefore $a_{kl.parent}$ will learn this secret in the $a_{kl}a_{kl.parent}$ call. We can iterate the same argument for all the calls

that are generated by function Bubble, since in each call an agent calls its parent node after learning the secret of $a_{ch}$ from one of its child nodes, until the root node learns this secret too. Hence we see that the calling condition for Known Information Growth de Re is satisfied for each of the calls in function Bubble.

3. Now, after the execution of the Bubble function, let us consider what happens in Lines 11-16 of function BFP. We see that beginning from the root node, until node $a_{kl}$, each node calls all its child nodes, except for (a) calls that are the reverse call of one of the calls made during function Bubble, *and* (b) node $a_{kl}$ which calls only the child nodes whose index is less than that of $a_{ch}$. We see that, beginning from the root node, every such call between a parent node $a_i$ and its child node $a_j$, the secret of $a_{ch}$ is learnt by $a_j$. Now consider the node $a_{ch}$ and such node $a_j$, these two nodes form the terminals of a line topology network. Hence prior to the $a_ia_j$ call, by Property 6.15' (or Property 6.15 as the case may be), $a_i$ knows that $a_j$ will learn the secret of $a_{ch}$ in the call.

From 1 - 3 we see that the calling condition for Known Information Growth de Re is satisfied for each of the calls in the call sequence generated by Procedure 6.3, and hence the generated call sequence is an execution sequence of Known Information Growth de Re. □

**Proposition 6.51.** *The execution sequence generated by Procedure 6.3 is successful.*

*Proof.* Consider Line 7 of Procedure 6.3. For any node $a_{kl}$, and for any child node $a_{ch}$ of $a_{kl}$, let the set of agents $\mathsf{Ag}'$ be defined as: $\mathsf{Ag}' = \{a_{ij} |\ ij < ch\}$. (Observe that $\mathsf{Ag}'$ is the set of nodes that are considered during function BFP). We first show that just before executing Line 7 of Procedure 6.3 (just after Line 6), every $a_i \in \mathsf{Ag}'$ knows only the secret of every agent in $\mathsf{Ag}'$, and we will conclude by using this property to show that the generated execution sequence is successful. We proceed by induction over the index $kl$ of nodes.

**Base Case**

Initially, for $a_{kl} = a_{01}$, and for the child node $a_{11}$ of $a_{01}$, $\mathsf{Ag}' = \{a_{01}\}$, i.e. the set $\mathsf{Ag}'$ comprises only the root node, so indeed every agent in $\mathsf{Ag}'$ knows only the secret of every agent in $\mathsf{Ag}'$, namely, the secret of $a_{01}$.

**Inductive Hypothesis**

As the inductive hypothesis, suppose that for all $a_{kl}$ such that $kl \leq (h-1)t$ (where $t$ is the number of nodes in layer $h-1$), and for all child nodes $a_{xy}$ of $a_{kl}$ where $xy < hw$, it is the case that every $a_i \in \mathsf{Ag}'$ knows only the secret of every agent in $\mathsf{Ag}'$, just before executing Line 7 of Procedure 6.3. (Notice from Figure 6.7 that $a_{hw}$ is the node with the highest node index of the tree topology network i.e. the leftmost leaf node of the tree; so the inductive hypothesis considers all parent nodes in the tree, and considers all child nodes of all parent nodes, except the very last child node in order of increasing node index, namely $a_{hw}$).

**Inductive Step**

Now consider the set $\mathsf{Ag}'$ for the node $a_{(h-1)t}$ and the child node $a_{hw}$, where $a_{hw}$ is the child node of $a_{(h-1)t}$ (again, $a_{hw}$ is the node with the highest node index of the tree topology network). Notice also that $a_{(h-1)t} \in \mathsf{Ag}'$, and from the inductive hypothesis above, the only secret that is not known by any agent in $\mathsf{Ag}'$ is the secret of $a_{hw}$, since $a_{hw}$ is currently the only node that is not in $\mathsf{Ag}'$. Now, Line 7 involves agent $a_{(h-1)t}$ calling agent $a_{hw} \notin \mathsf{Ag}'$. Notice that this call $a_{(h-1)t}a_{hw}$ is the last call in the execution sequence generated by Procedure 6.3. Since $a_{hw}$ is a fresh caller in such $a_{(h-1)t}a_{hw}$ call, then $a_{(h-1)t}$ learns only the secret of $a_{hw}$ in that call. After that call, $a_{(h-1)t}$ knows one more secret, namely, that of $a_{hw}$, than any other agents in $\mathsf{Ag}'$. Also, agent $a_{hw}$ learns the secret of every agent in $\mathsf{Ag}'$ from that call. Hence after the $a_{(h-1)t}a_{hw}$ call, both agents know all the secrets in the scenario. Now observe that in the execution of function BFP on Line 8, every call generated by BFP is between an agent who knows $a_{ch}$ and an agent who does not know $a_{ch}$. Furthermore, in BFP, every agent in $\mathsf{Ag}'$ takes part in at least one call. Therefore after BFP, every agent in $\mathsf{Ag}'$ knows the secret of $a_{ch}$. Hence when function BFP exits, it is clear then that every agent in $\mathsf{Ag}'$ knows all the secrets. And thus, every agent has now learnt all the secrets in the scenario. Finally, the generated execution sequence terminates, since at this point the conditions for the 'for' loops can no longer be satisfied, since the last node index on the tree has been considered and the height of the tree exhausted. □

**Proposition 6.52.** *Let* $\Pi$ *be a regular gossip protocol, and let* $\Pi$ *be successful. Then no execution sequence of* $\Pi$ *is longer than the execution sequence generated by Procedure 6.3.*

*Proof.* Let the execution sequence generated by Procedure 6.3 be $\sigma^*$. Consider Procedure 6.3. Let us for now ignore the call to function BFP on Line 8, and let us refer now to what remains of this procedures as *the downward movement*. Now notice that the downward movement yields only fresh calls. Beginning from layer 0, until layer $h-1$, each node calls each of all its child nodes exactly once, and in each such call, the child node is a fresh caller. Again, the total number of the resulting calls is $n-1$, corresponding to a use of each of the $n-1$ edges of the tree topology network.

Now all other calls that we have not considered so far are generated from function BFP. We now show that in each of the calls generated by function BFP, the total number of secrets learnt by the calling pair is one.

- Consider Line 7 of Procedure 6.3. For any node $a_{kl}$, and for any child node $a_{ch}$ of $a_{kl}$, let the set of agents $\mathsf{Ag}'$ be defined as: $\mathsf{Ag}' = \{a_{ij} |\ ij < ch\}$. (Observe that $\mathsf{Ag}'$ is the set of nodes that are considered during function BFP). We first show that just before executing Line 7 of Procedure 6.3 (just after Line 6), every $a_i \in \mathsf{Ag}'$ knows the same set of secrets. We proceed by induction over the index $kl$ of nodes.

**Base Case**

Initially, for $a_{kl} = a_{01}$, and for the child node $a_{11}$ of $a_{01}$, $\mathsf{Ag}' = \{a_{01}\}$, i.e. the set $\mathsf{Ag}'$ comprises only the root node, so indeed every agent in $\mathsf{Ag}'$ knows the same set of secrets, namely the secret of $a_{01}$.

**Inductive Hypothesis**

Suppose that for all $a_{kl}$ such that $kl < st$, and for all child nodes of $a_{kl}$, it is the case that every $a_i \in \mathsf{Ag}'$ knows the same set of secrets, just before executing Line 7 of Procedure 6.3.

**Inductive Step**

Now consider the set $\mathsf{Ag}'$ for the node $a_{st}$ and a child node $a_{ch}$ of $a_{st}$. (Notice that $a_{st} \in \mathsf{Ag}'$). Now, Line 7 involves agent $a_{st} \in \mathsf{Ag}'$ calling agent $a_{ch} \notin \mathsf{Ag}'$. Since $a_{ch}$ is a fresh caller in such $a_{st}a_{ch}$ call, then $a_{st}$ learns only the secret of $a_{ch}$ in that call. After that call, $a_{st}$ knows one more secret, namely, that of $a_{ch}$, than any other agents in $\mathsf{Ag}'$. Now observe that in the execution of function BFP on Line 8, every call generated by BFP is between an agent who knows the secret of $a_{ch}$ and an agent who does not know the secret of $a_{ch}$. Furthermore, in BFP, every agent in $\mathsf{Ag}'$ takes part in at least one call. Therefore after BFP, every agent in $\mathsf{Ag}'$ knows the secret of $a_{ch}$. When function BFP exits, and the 'for' loop at Line 6 begins again, it is clear then that the property that every agent in $\mathsf{Ag}'$ knows the same set of secrets is maintained.

- Now, since just before Line 7 the secrets known by all the agents in $\mathsf{Ag}'$ are the same, and after the call in Line 7 the agent $a_{kl}$ knows one more secret (namely, the secret of $a_{ch}$) than other agents in $\mathsf{Ag}'$, and every call in BFP is between an agent who knows the secret of $a_{ch}$ and an agent who does not know the secret of $a_{ch}$. Then in every call in BFP, the total number of secrets learnt by the calling pair is exactly one, namely, the secret of $a_{ch}$. Since in BFP, the total number of calls is $(n-1)(n-2)/2$ calls, then there are $(n-1)(n-2)/2$ calls in which the total number of secrets learnt is exactly one.

Now recall that the downward movement comprises only of fresh calls, and there are $n-1$ such calls. In each of those calls, the total number of secrets learnt per call is at least two. Recall also that in every successful execution sequence there are at most $n-1$ fresh calls (from Lemma 6.10). Furthermore, for a regular protocol, such successful execution sequence must have the property that in each call at least one secret is learnt by some agent in the call. But from Proposition 6.50 we know that $\sigma^*$ is an execution sequence of Known Information Growth de Re, and, from Proposition 6.9, we know that Known Information Growth de Re is regular. Also, $\sigma^*$ is successful (from Proposition 6.51), therefore it must have at most $n-1$ fresh calls. Now, from the foregoing analysis we see that the execution sequence generated by Procedure 6.3 has exactly $n-1$ fresh calls, and it has $(n-2)(n-1)/2$ calls in which the total number of secrets learnt is

exactly one. Hence we see that no other execution sequence of a regular and successful protocol can be made longer than $\sigma^*$, unless such an execution sequence contained some call in which the total number of secrets learnt is zero, which will then contradict the assumption that the protocol is regular. □

**Corollary 6.53.** *Given a tree topology network of agents, then the maximum length of an execution sequence in the extension of a regular and successful epistemic gossip protocol is* $n(n-1)/2$.

*Proof.* The proof follows from the proof of Proposition 6.52 in which we show that the length of the execution sequence generated by Procedure 6.3 is the maximum for any regular and successful gossip protocol on a tree topology network. □

**Corollary 6.54.** *Given a tree topology network of agents, then the maximum length of an execution sequence of Known Information Growth de Re and Known Information Growth de Dicto is* $n(n-1)/2$.

*Proof.* From Corollary 6.53 we see that the length of the execution sequence generated by Procedure 6.3 is the maximum for any regular and successful gossip protocol on the tree topology network. In particular, the length of the execution sequence generated by Procedure 6.3 is the maximum for Known Information Growth de Re (from Proposition 6.50). Finally, the length of the execution sequence generated by Procedure 6.3 is the maximum for Known Information Growth de Dicto (from Proposition 3.27 and Observation 4.26: every execution sequence of Known Information Growth de Re is an execution sequence of Known Information Growth de Dicto). □

### 6.4.2 Synthesising Minimum Length Successful Execution Sequences on a Tree Topology Network

We now consider the shortest successful execution sequence length for Known Information Growth de Dicto and Known Information Growth de Re, on a tree topology network. In their paper [28], Harary and Schwenk gave a procedure for obtaining a shortest successful execution sequence on an arbitrary connected (but incomplete) network graph and they proved that such an execution sequence is of length $2n-3$. The procedure is based on a spanning tree of an arbitrary connected network graph, and proceeds in two stages, namely, an *upward movement* and a *downward movement*.

**Upward Movement** The upward movement is as follows. Let each of all the leaf nodes call its parent node in turn. Now remove all the current leaf nodes and their adjoining edges so that their parent nodes become the new leaf nodes of the tree. Repeat the process of each of the current leaf nodes calling its parent and the removal of the current leaf nodes, until only the root node is remaining.

**Downward Movement** Now we do the downward movement as follows. Replace[‡] all the child nodes of the root node and their adjoining edges. Except for the last node that called the root node in the upward movement, let the root node call all its child nodes in turn. The child nodes are at a distance $\delta = 1$ from the root node. So likewise, iteratively replace the child nodes (and their adjoining edges) which were at a distance $\delta = 2, 3, \ldots h$ in the original tree, where $h$ is the height of the tree. In each iteration let each of all the nodes at a distance $\delta - 1$ call its child nodes.

Note that only a pair of nodes can call at a time. Let each call between two nodes be considered as a use of the edge between two such nodes. In the upward movement, each edge of the tree is used only once, and this gives a total of $n - 1$ calls. In the downward movement, the total number of uses of the edges is $n-2$ since the last call in the upward movement is not repeated in the downward movement. So now we have a total of $2n-3$ calls. Suppose any edge (or call) were to be omitted from the upward movement. Let the edge in question be between some node $\eta_1$ at a distance $\delta$ and another node $\eta_2$ at a distance $\delta + 1$ from the root node. Then all the nodes at a distance $\delta' \leq \delta$ will never learn the secrets known by the node $\eta_2$. The situation is analogous for the downward movement if such an edge were omitted from it. Since all the $2n-3$ uses of the edges (or calls) are necessary, the conclusion is that we cannot have a shorter execution sequence than $2n - 3$ on such a network graph.

We show that the procedure described above as an upward and downward movement, yields an execution sequence of Known Information Growth de Re (and therefore also an execution sequence of Known Information Growth de Dicto – see Proposition 3.27 and Observation 4.26) on a tree topology network. Recall that we make the assumption that both the protocol and the network topology are common knowledge among all the agents in the scenario, so each agent knows whom can call who as far as the network links are concerned. In each call $a_i a_j$ from node $a_i$ to node $a_j$ in the upward movement, node $a_i$ knows that node $a_j$ will learn the secret of node $a_i$ in the $a_i a_j$ call. Hence the calling condition for Known Information Growth de Re is satisfied for each call in the upward movement. In the downward movement where each node calls its child nodes, in each $a_i a_j$ call, node $a_i$ knows that node $a_j$ will learn secret $A_k$, where $A_k$ is the secret that $a_i$ learnt in the call immediately preceding the $a_i a_j$ call (node $a_i$ knows this since it knows that its child nodes can only learn secret $A_k$ through a call with it). Hence, once more the calling condition for Known Information Growth de Re is satisfied for each call in the downward movement. As a result, we conclude that on a tree network graph, the shortest successful execution sequence for Known Information Growth de Dicto and Known Information Growth de Re protocol is of length $2n - 3$.

[‡]We use the word "replace" here with the understanding that the downward movement on the tree is executed after the upward movement on the same tree.

**Altering the Tree Topology Network for a Lower Minimum Length of Successful Execution Sequences**

From the preceding subsection, a follow-up discussion is about whether there is a way to alter the tree topology network so as to reduce the length of the shortest successful execution sequence of the Known Information Growth protocols from $2n-3$ to the shortest length obtainable in a complete topology network, namely, $2n-4$. The answer to this question is yes. Given a connected topology network of $n$ agents with only one cycle, which is a 4-cycle, the shortest successful execution sequence of Known Information Growth de Dicto and Known Information Growth de Re is of length $2n-4$. To demonstrate this claim consider Figure 6.11 which depicts a connected topology network with one cycle, which is a 4-cycle (the network is in fact a tree topology network with one tree node replaced by a 4-cycle). We consider the network as having two parts, namely, a tree part $\tau$ and a 4-cycle part $\omega$, as shown in Figure 6.11.

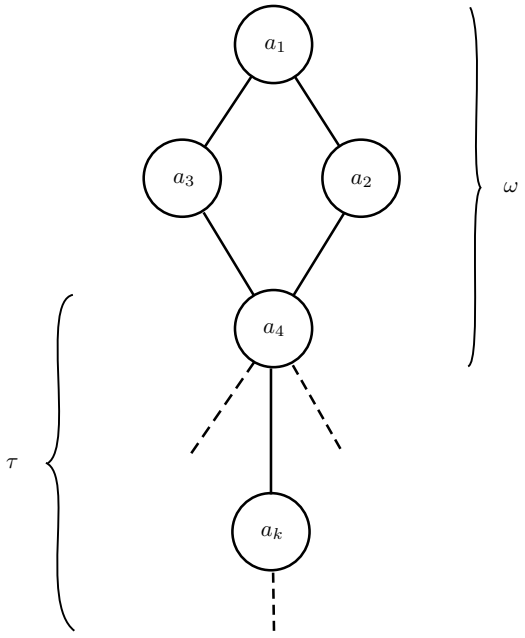



FIGURE 6.11: **A connected topology network with one 4-Cycle.**

Now, let the *upward movement* be as described in the preceding section, but now applied to the tree $\tau$ in Figure 6.11. We modify the *downward movement* given in the preceding section to get a new downward movement, as follows (notice that in the modified downward movement, we now allow the root node of the tree $\tau$ to call the last node that called it in the upward movement):

**Modified Downward Movement** Consider the tree $\tau$ in Figure 6.11. Replace[§] all the child nodes of the root node and their adjoining edges. Let the root node call all

[§]We use the word "replace" here with the understanding that the modified downward movement on the tree is executed after the upward movement on the same tree.

its child nodes in turn. The child nodes are at a distance $\delta = 1$ from the root node. So likewise, iteratively replace the child nodes (and their adjoining edges) which were at a distance $\delta = 2, 3, \ldots h$ in the original tree, where $h$ is the height of the tree. In each iteration let each of all the nodes at a distance $\delta - 1$ call its child nodes.

Now let the following procedure be carried out, Step 1 through Step 3.

*Procedure* 5.2.

Step 1 Execute the upward movement as described in the preceding subsection, on the tree $\tau$.

Step 2 Let the agents on the 4-cycle $\omega$ call each other according to the *Four-Agent Protocol* as follows: any two agents make the first call; the second call is then between the remaining two agents; the third call is then between an agent who made the first call and an agent who made the second call; and the fourth call is between the two who were not chosen in the third call.

Step 3 Execute the modified downward movement.

Now let the number of nodes on the tree $\tau$ be $m$ (note that the root of the tree $\tau$ is the node $a_4$, as shown in Figure 6.11; note too that node $a_4$ is also part of the 4-cycle $\omega$). Now the total number of nodes on the network is given by $n = m + 3$.

Observe that the *modified downward movement* has one more call than the *downward movement*. Following the analysis in the preceding subsection, the total number of calls due to the upward movement and the modified downward movement is then given by $2(m - 1)$. The *Four-Agent Protocol* clearly gives four calls. Therefore the total number of calls due to the foregoing procedure is $2m - 2 + 4 = 2(n - 3) + 2 = 2n - 4$.

Finally, the analysis which demonstrates that Procedure 5.2 yields execution sequences of Known Information Growth de Dicto and Known Information Growth de Re, is the same as the analysis given in the preceding section for a similar purpose, but with one exception. The only difference is that here we additionally justify the extra call in the modified downward movement, namely, the call from the root node of $\tau$, namely $a_4$, to the agent who called it last in the upward movement. Let the agent who called $a_4$ last in the upward movement be $a_i$. The call $a_4a_i$ is allowed in the modified downward movement by Known Information Growth de Re (and consequently by Known Information Growth de Dicto – see Proposition 3.27 and Observation 4.26). The reason is that after the last call of the *Four-Agent Protocol* (Step 2 of Procedure 5.2), agent $a_4$ knows that $a_i$ does not yet know the secret of any of the agents in the 4-cycle $\omega$. In particular, agent $a_4$ knows that in the call $a_4a_i$ in the modified downward movement, agent $a_i$ will learn the secret of agent $a_2$, thus the calling condition for the $a_4a_i$ call is satisfied for Known Information Growth de Re (and also for Known Information Growth de Dicto).

## 6.5 Circle Topology Network

In this section we consider a gossip scenario in which the network of agents is a circle topology network of $n$ agents, and consider the properties of Learn New Secrets, Known Information Growth de Dicto, Known Information Growth de Re, Possible Information Growth de Dicto and Possible Information Growth de Re.

Let the arrangement of the agents be as shown in Figure 6.12:

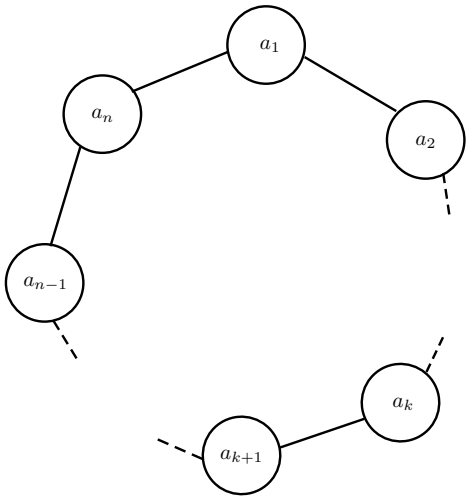



FIGURE 6.12: **Circle Topology Network.**

**Proposition 6.55.** *Learn New Secrets protocol is not successful for a gossip scenario where the network of agents is a circle topology network and the size n of the set of agents is greater than three.*

*Proof.* Recall that the calling condition for agent $a_i$ to call agent $a_j$ under Learn New Secrets protocol is: that agent $a_i$ does not know the unique secret of agent $a_j$, and, agent $a_i$ and $a_j$ must be neighbours on the network graph.

From the circle topology network of agents shown in Figure 6.12, consider the following execution sequence $\sigma$. Observe that $\sigma$ is an execution sequence of Learn New Secrets on the circle topology network in Figure 6.12, and therefore $\sigma$ is in the extension of Learn New Secrets for that circle topology network of agents, where the size of the set of agents is greater than three:

$$\sigma \quad = \quad a_1a_2; a_2a_3; \ldots; a_{k-1}a_k; \ldots; a_{n-1}a_n \quad \text{(for all } 3 < k < n\text{)}$$

Now, after the last call of $\sigma$, agent $a_{n-1}$ and agent $a_n$ know all the secrets in the scenario, but this is not the case for the other agents. Moreover, no other agents can make any more calls since each agent has learnt the secret of all its neighbours. □

We now consider the Known Information Growth de Dicto and Known Information Growth de Re protocols. In order to show that both Known Information Growth de Re

and Known Information Growth de Dicto protocols can deadlock on a circle topology network, we work through an example of a deadlocked execution sequence of both protocols in a gossip scenario comprising of five agents $(a_1, a_2, a_3, a_4, a_5)$ connected in a circle topology network (see Figure 6.12). Furthermore, for a more intuitive exposition, we choose to adopt the equivalence notion given in Definition 4.40 for this example. Recall that for the equivalence notion given in Definition 4.40 an agent considers what secrets it learnt from each call, in a pair of histories, in order to determine whether such pair of histories are equivalent. Now, consider the following execution sequence $\sigma$:

$$\sigma \quad = \quad a_1a_2; a_1a_5; a_2a_3; a_5a_4; a_3a_4; a_5a_1; a_2a_3; a_1a_2$$

We show that $\sigma$ is deadlocked under Known Information Growth de Dicto on the circle topology network, as follows. At $\sigma$ the only agent who does not yet know all the secrets in the scenario is agent $a_5$. Therefore under Known Information Growth de Dicto (and Known Information Growth de Re), the only calls that can be considered at $\sigma$ on the circle topology network are calls between agent $a_5$ and any one of its neighbours. Hence we consider the following calls at $\sigma$: $a_5a_1$, $a_1a_5$, $a_5a_4$ and $a_4a_5$.

To show that the call $a_5a_1$ is not possible at $\sigma$, consider the execution sequence $\sigma_1$ as follows:

$$\sigma_1 \quad = \quad a_1a_2; a_1a_5; a_1a_2; a_5a_4; a_2a_3; a_5a_1; a_3a_4; a_2a_3$$

For agent $a_5$, the execution sequence $\sigma$ is equivalent to $\sigma_1$ under the definition of equivalent histories given in Definition 4.40 (the reader can verify this following the definition of equivalent histories given in Definition 4.40). At $\sigma_1$, both agent $a_5$ and agent $a_1$ know the same set of secrets. Therefore at $\sigma$, agent $a_5$ considers it possible that there is no secret that will be learnt in an $a_5a_1$ call.

To show that the call $a_1a_5$ is not possible at $\sigma$, consider the execution sequence $\sigma_2$ as follows:

$$\sigma_2 \quad = \quad a_1a_2; a_1a_5; a_2a_3; a_5a_4; a_3a_4; a_5a_1; a_4a_5; a_1a_2$$

For agent $a_1$, the execution sequence $\sigma$ is equivalent to $\sigma_2$ under the definition of equivalent histories given in Definition 4.40. At $\sigma_2$, both agent $a_1$ and agent $a_5$ know all the secrets in the scenario. Therefore at $\sigma$, agent $a_1$ considers it possible that there is no secret that will be learnt in an $a_1a_5$ call.

To show that the call $a_5a_4$ is not possible at $\sigma$, consider the execution sequence $\sigma_3$ as follows:

$$\sigma_3 \quad = \quad a_1a_2; a_1a_5; a_1a_2; a_5a_4; a_2a_3; a_5a_1; a_1a_2; a_2a_3$$

For agent $a_5$, the execution sequence $\sigma$ is equivalent to $\sigma_3$ under the definition of equivalent histories given in Definition 4.40. At $\sigma_3$, both agent $a_5$ and agent $a_4$ know the same set of secrets. Therefore at $\sigma$, agent $a_5$ considers it possible that there is no secret that will be learnt in an $a_5a_4$ call.

To show that the call $a_4a_5$ is not possible at $\sigma$, consider the execution sequence $\sigma_4$ as follows:

$$\sigma_4 \quad = \quad a_1a_2; a_1a_5; a_2a_3; a_5a_4; a_3a_4; a_2a_1; a_1a_5; a_1a_2$$

For agent $a_4$, the execution sequence $\sigma$ is equivalent to $\sigma_4$ under the definition of equivalent histories given in Definition 4.40. At $\sigma_4$, both agent $a_4$ and agent $a_5$ know all the secrets in the scenario. Therefore at $\sigma$, agent $a_4$ considers it possible that there is no secret that will be learnt in an $a_4a_5$ call.

Since none of the calls $a_1a_5$, $a_5a_1$, $a_4a_5$ or $a_5a_4$ is possible at $\sigma$, the execution sequence $\sigma$ is deadlocked under Known Information Growth de Dicto. It is also the case that the execution sequence $\sigma$ is deadlocked under Known Information Growth de Re. To see this, recall that at $\sigma$, the only agent who does not yet know all the secrets is agent $a_5$. Since at $\sigma$ agent $a_5$ knows all the secrets except only that of agent $a_3$, it follows that if an agent knows that some secret will be learnt in the next call after $\sigma$, then that agent knows that it is agent $a_5$ who will learn the secret of agent $a_3$ in that next call (here, epistemic calling condition for Known Information Growth de Dicto implies that for Known Information Growth de Re). On the other hand, if an agent knows that the secret of agent $a_3$ will be learnt in the next call after $\sigma$, then that agent knows that it is agent $a_5$ who will learn some secret (namely, that of agent $a_3$) in that call (here, the epistemic calling condition for Known Information Growth de Re implies that for Known Information Growth de Dicto). Therefore, the epistemic calling condition for Known Information Growth de Dicto is equivalent to that for Known Information Growth de Re for the next call after $\sigma$. Hence if $\sigma$ is deadlocked under Known Information Growth de Dicto, then it is also deadlocked under Known Information Growth de Re. And if $\sigma$ is deadlocked under Known Information Growth de Re, then it is also deadlocked under Known Information Growth de Dicto.

Now, let us demonstrate that the execution sequence $\sigma$ is actually an execution sequence of Known Information Growth de Dicto. To show that $\sigma$ is an execution sequence of Known Information Growth de Dicto it is enough to show that $\sigma$ is an execution sequence of Known Information Growth de Re (from Proposition 3.27, an execution sequence of Known Information Growth de Re is also an execution sequence of Known Information Growth de Dicto).

To see that $\sigma$ is an execution sequence of Known Information Growth de Re, consider the following. In each of the first five calls in $\sigma$, the agent who initiates the call knows that it will learn the secret of the callee agent, since in each call the agent who initiates the call did not know the secret of the callee agent prior to the call (for example, for the first call of $\sigma$, agent $a_1$ knew that it will learn the secret of agent $a_2$; for the second call agent $a_1$ knew that it will learn the secret of agent $a_5$; etc.). So for the first five calls in $\sigma$ it is clear that the agent who initiates the call knows of a secret that will be learnt in the call by either of the agents in the call.

For the sixth call in $\sigma$, consider that in the fourth call in $\sigma$ agent $a_5$ learnt the secret of agent $a_4$, but agent $a_5$ also learnt that that call was the first call to be made by agent

$a_4$ in the execution sequence (this is due to the set of secrets which agent $a_5$ learnt from that $a_5a_4$ call), so agent $a_5$ knows that after that fourth call, without a call between $a_5$ and $a_1$, a minimum of three more calls is required for agent $a_1$ to learn the secret of agent $a_4$ (due to further calls $\ldots a_4a_3; a_3a_2; a_2a_1; \ldots$), so for the sixth call agent $a_5$ knows that agent $a_1$ does not yet know the secret of agent $a_4$, since $a_5$ knows the fifth call was not between $a_5$ and $a_1$. And since agent $a_5$ knows the secret of agent $a_4$, the calling condition for Known Information Growth de Re is satisfied for an $a_5a_1$ call in the sixth call in $\sigma$.

For the seventh call $a_2a_3$ in $\sigma$, consider that agent $a_2$ called agent $a_3$ in the third call and learnt that that was the first call agent $a_3$ has made. Also, agent $a_2$ was in the first call in $\sigma$ and therefore knows that the first call was with $a_1$. Therefore after the third call in $\sigma$, agent $a_2$ knows that the second call in $\sigma$ must have been either between agent $a_4$ and agent $a_5$, or between agent $a_1$ and agent $a_5$, given the arrangement of agents on the circle topology network (see Figure 6.12).

Given that agent $a_2$ was not involved in the fourth, fifth and sixth calls in $\sigma$ (and therefore none of those calls was between $a_1$ and $a_2$ or between $a_2$ and $a_3$):

(i) Suppose the second call in $\sigma$ was between $a_4$ and $a_5$, then the fourth call is either a call between $a_1$ and $a_5$, or a call between $a_3$ and $a_4$. If the fourth call was between $a_1$ and $a_5$, then the fifth call will either be between $a_4$ and $a_5$, or between $a_3$ and $a_4$. If the fifth call was between $a_4$ and $a_5$, and if prior to the fifth call $a_3$ and $a_4$ have not called each other, then the sixth call cannot be again between $a_1$ and $a_5$ since none of them will learn any new secret from such call, so in this case the sixth call must be between $a_3$ and $a_4$. Therefore, if the second call in $\sigma$ was between $a_4$ and $a_5$, then $a_3$ and $a_4$ must have called each other in at least a call between the fourth and sixth call inclusive, in $\sigma$.

(ii) Suppose the second call was between $a_1$ and $a_5$, then the fourth call is either between $a_4$ and $a_5$ or between $a_3$ and $a_4$. If the fourth call was between $a_4$ and $a_5$, then the fifth call is either between $a_1$ and $a_5$, or between $a_3$ and $a_4$. If the fifth call was between $a_1$ and $a_5$, and prior to the fifth call $a_3$ and $a_4$ have not called each other, then the sixth call cannot be between $a_4$ and $a_5$ again since then both agents will have no new secret to learn from each other, so in this case the sixth call must be between $a_3$ and $a_4$. Therefore, if the second call in $\sigma$ was between $a_1$ and $a_5$, then $a_3$ and $a_4$ must have called each other in at least a call between the fourth and sixth call inclusive, in $\sigma$.

Given that both the protocol and the topology of the underlying network are common knowledge among all the agents, and from the foregoing (i) and (ii) cases, it follows that in all execution sequences that are equivalent to $\sigma$ for agent $a_2$, it is the case that $a_3$ and $a_4$ called each other at least once between the fourth call and the sixth call inclusive. Therefore in the seventh call in $\sigma$, agent $a_2$ calls $a_3$ knowing that it will learn the secret of agent $a_4$ from agent $a_3$ in that call.

For the eighth call $a_1a_2$ in $\sigma$, to see that $a_1$ knows that it will learn the secret of $a_3$ from $a_2$, consider the following. In the second call of $\sigma$, agent $a_1$ called agent $a_5$ and learnt that that was the first call $a_5$ made in the execution sequence. In that second call, $a_5$ learnt the secret of agent $a_1$, and the secret of agent $a_2$. In the sixth call in $\sigma$, agent $a_5$ and agent $a_1$ called each other for the second time in the execution sequence, and there $a_1$ learnt that $a_5$ had learnt only the secret of agent $a_4$ since their first ($a_1a_5$) call with each other, that is, in the second call in $\sigma$. So at the sixth call in $\sigma$, agent $a_1$ knows that agent $a_5$ must have made a call with only agent $a_4$ between calls three and five inclusive, in $\sigma$. Furthermore, after the sixth call in $\sigma$, agent $a_1$ knows that the two other calls (that is, the calls apart from that between $a_4$ and $a_5$) from calls three to five did not involve both itself ($a_1$) and $a_5$, and so one of those two other calls (from calls three to five) must have been between $a_2$ and $a_3$. Therefore, at the eighth call in $\sigma$, agent $a_1$ knows that agent $a_2$ knows the secret of agent $a_3$, hence agent $a_1$ knows that it will learn the secret of agent $a_3$ in the eighth call $a_1a_2$.

We have now shown that $\sigma$ is an execution sequence of Known Information Growth de Re, and therefore also an execution sequence of Known Information Growth de Dicto. Similar reasoning for $\sigma$ can be used to show that $\sigma_1$, $\sigma_2$, $\sigma_3$ and $\sigma_4$ are execution sequences of Known Information Growth de Re, and therefore also execution sequences of Known Information Growth de Dicto.

**Proposition 6.56.** *Possible Information Growth de Re is non-terminating in a gossip scenario where the network of agents is a circle topology network and the size n of the set of agents is greater than three.*

*Proof.* Recall that the calling condition for Possible Information Growth de Re states that for any agent $a_i$ to call another agent $a_j$, there must be a secret $A_k$ such that $a_i$ considers it possible that one of $a_i$ and $a_j$ will learn $A_k$ from the $a_ia_j$ call; moreover, agent $a_i$ and $a_j$ must be neighbours on the network graph.

From the circle topology network of agents shown in Figure 6.12, consider the following execution sequence $\sigma$ for a gossip scenario where $n > 3$.

$$\sigma \quad = \quad a_1a_2; a_3a_4; a_1a_2; a_4a_3; a_1a_2; a_4a_3 \ldots$$

We begin by demonstrating that $\sigma$ is an execution sequence of Possible Information Growth de Re, and therefore it is in the extension of the protocol. And then we show that $\sigma$ is non-terminating.

In the first call in $\sigma$, agent $a_1$ knows (and therefore considers it possible) that it would learn the secret of agent $a_2$ in the first call. And likewise, in the second call, agent $a_3$ knows that it will learn the secret of agent $a_4$ in the second call with agent $a_4$. In the third call, agent $a_1$ considers it possible that agent $a_2$ learnt some secret in the second call (in a possible call with agent $a_3$), and so agent $a_1$ calls agent $a_2$ in the third call, which proves redundant. Likewise in the fourth call, agent $a_4$ considers it possible that agent $a_3$ learnt some secret in the third call (in a possible call with agent

$a_2$) and therefore agent $a_4$ calls agent $a_3$ in the fourth call which also proves redundant. Once again agent $a_1$ considers it possible that agent $a_2$ learnt some new secret in the preceding (fourth) call, and thus for the execution sequence $\sigma$, the loop $a_1a_2; a_4a_3; \ldots$ goes on infinitely. □

**Proposition 6.57.** *Possible Information Growth de Dicto is non-terminating in a gossip scenario in which the network of agents is a circle topology network and the size n of the set of agents is greater than three.*

*Proof.* The proof follows from the proof of Proposition 6.56, and from the proof of Proposition 3.27 (and Observation 4.26) which shows that the extension of Possible Information Growth de Re is equal to the extension of Possible Information Growth de Dicto. □

**Proposition 6.58.** *Possible Information Growth de Dicto and Possible Information Growth de Re are not successful in a gossip scenario in which the network of agents is a circle topology network and the size n of the set of agent is greater than three.*

*Proof.* The proof follows from the proof of Proposition 6.56 and 6.57, and from the definition of a successful protocol as given in Definition 4.20. □

Some follow-up problems to the foregoing are: (a) What is the minimum number of edges that can be added to a circle topology network with more than four agents such that Known Information Growth de Re is successful on the resulting network? (b) What is the minimum number of edges that can be added to a circle topology network with more than four agents such that Known Information Growth de Dicto is successful on the resulting network?

**Conjectures** Finally, before we close this chapter, we present the following two conjectures about epistemic gossip protocols.

*Conjecture* 6.59. The reverse of a successful execution sequence of an epistemic gossip protocol is also a successful execution sequence.

*Conjecture* 6.60. The reverse of a successful execution sequence of the Learn New Secrets protocol is also a successful execution sequence of the Learn New Secrets protocol.

## 6.6 Conclusion

In this chapter we studied some theoretical properties of our epistemic gossip protocols on various network topologies. We studied the complete, circle and tree topology networks (including a special treatment of the line topology network), and we presented proofs of properties of Learn New Secrets, Known Information Growth de Dicto, Known Information Growth de Re, Possible Information Growth de Dicto and Possible Information Growth de Re for these various network topologies.

Our results show that for $n$ agents, Known Information Growth de Re and Known Information Growth de Dicto are successful on a complete topology network, and on any tree topology network. On the other hand, Learn New Secrets is only successful on a complete topology network, for $n$ agents, whereas Possible Information Growth de Dicto and Possible Information Growth de Re are shown to be unsuccessful for complete and tree topology networks with more than three agents.

We showed that the maximum length of an execution sequence of a regular and successful gossip protocol is $n(n-1)/2$. In particular, this result applies to Learn New Secrets, Known Information Growth de Re and Known Information Growth de Dicto, since these protocols are regular. On the other hand, we showed also that the length of the shortest successful execution sequence of Known Information Growth de Re and Known Information Growth de Dicto is $2n-4$, if the underlying network is a complete topology network. But, for the tree topology network we showed that the length of the shortest successful execution sequence of Known Information Growth de Dicto and Known Information Growth de Re is $2n-3$. We further showed that for a connected topology network with only one cycle, and where the cycle is a 4-cycle, the length of the shortest successful execution sequence of Known Information Growth de Re and Known Information Growth de Dicto is $2n-4$. The foregoing results for the lengths of shortest successful execution sequences are the same as the results in the traditional gossip literature [30] for non-epistemic settings of the gossip problem.

Other possible directions for future work from this chapter are as follows: the study of other network topologies such as grid networks; synthesis of epistemic gossip protocols that yield only the shortest length execution sequences for a given network topology; synthesis of epistemic gossip protocols that yield only the longest length execution sequences for a given network topology; questions regarding the minimum number of edges to add to a circle topology network so as to ensure that the Known Information Growth protocols, for example, are successful on the resulting network; and questions regarding epistemic gossip protocols that are successful on a circle topology network.

# Chapter 7

# Conclusion

In this thesis we developed a framework based on dynamic epistemic logic to study distributed epistemic gossip protocols*. We described such protocols and studied them from both a theoretical and empirical point of view. We now give a more detailed summary of the contributions of this thesis, and provide pointers for future research.

## 7.1 Summary of Contributions

We recall the research questions posed in Chapter 1, as follows.

**Research Question 1:** Is it possible to describe epistemic gossip protocols for the successful spreading of information in a network of autonomous agents?

**Research Question 2:** Is it possible to create, and use, some formalism based on dynamic epistemic logic to specify, model, analyse and verify epistemic gossip protocols?

**Research Question 3:** Is it possible to create a software framework to automate the empirical analysis of epistemic gossip protocols, given a specification of such a protocol?

**Research Question 4:** How does various network connectivity constraints affect the properties of epistemic gossip protocols?

Research Question 1 was answered in Chapters 3, 5 and 6. In Chapter 3, we described a number of epistemic gossip protocols. In Chapter 5 we studied the properties of these protocols from an empirical point of view, and in Chapter 6 we studied the properties of these protocols from an analytical point of view. Research Question 2 was answered in Chapters 3, 4 and 6. In Chapter 3, we presented a formalism which is based on dynamic epistemic logic, for reasoning about epistemic properties of agents in a gossip scenario. We extended our formalism in Chapter 4, a difference being that we then interpret

*Throughout this thesis, we use the terms 'distributed epistemic gossip protocol' and 'epistemic gossip protocol' interchangeably, since we always assume that each agent executes its own protocol.

epistemic formulas on a more abstract structure than that presented in Chapter 3. In both Chapter 3 and Chapter 6, we used our formalism to study the logical properties of epistemic gossip protocols. Research Question 3 was answered in Chapters 3, 4 and 5. In Chapter 4, we built upon the work in Chapter 3 by implementing a software framework to automate the empirical analysis of epistemic gossip protocols. In Chapter 5, we presented and discussed the empirical results obtained from the use of our software tool in the analysis of our epistemic gossip protocols. Finally, Research Question 4 was answered in Chapters 5 and 6 where we carried out empirical and analytical studies, respectively, to determine how various epistemic gossip protocols perform on various network topologies. We now give a summary of the main contributions of this thesis, according to the various chapters.

In Chapter 3, we described a theoretical framework for epistemic gossip protocols. We defined some epistemic gossip protocols, namely the Learn New Secret protocol wherein an agent calls another agent if it does not know the secret of that other agent; Known Information Growth de Dicto wherein an agent calls another agent if it knows that one of them will learn some secret in the call; Known Information Growth de Re wherein an agent calls another agent if it knows of a secret that will be learnt by either of them from the call; Possible Information Growth de Dicto wherein an agent calls another agent if it considers it possible that one of them will learn some secret in the call; and Possible Information Growth de Re wherein an agent calls another agent if there is a secret that it considers it possible that either of them will learn from the call. Next, we defined a formal logical language and semantics for describing and reasoning about epistemic gossip protocols. Our language and semantics are based on dynamic epistemic logic, and propositional dynamic logic. The semantics of our formal language is based on a Gossip Model, which we proved to be equivalent to a Kripke model, and in particular a special kind of an S5 model. Finally, we formalised our protocol descriptions and proved some of the logical properties of our protocols.

In Chapter 4, we described an implementation of the theoretical framework described in Chapter 3. The implemented tool EGP automates the analysis of epistemic gossip protocols. First, we introduced a high level programming language EGPL for describing epistemic gossip protocols. Then, we described an interpreter for this language, together with the model generator and model checker that form part of the tool. The EGP tool generates the gossip tree for a given protocol, from which we can deduce important dynamic properties such as successfulness, termination, average execution length, standard extension, and standard extension size of the given protocol. In order to enhance the performance of the model checker, we combined the standard top-down and bottom-up procedures for epistemic model checking, and introduced a caching mechanism in the layers of the gossip tree for the equivalence classes of the nodes. We showed that while the time complexity of our model checking algorithm remained quadratic in the size of the model (which is the same as that of the bottom-up approach employed in the CTL labelling algorithm), we achieved a better performance in practice with respect to the

epistemic gossip scenarios because of a high reduction in the number of nodes considered during model checking (for example, in one of our epistemic gossip protocols, there was an estimated 99.9996% reduction in the number of nodes considered at a layer of the gossip tree during the checking of the propositional part of the epistemic calling condition for the protocol; see subsection *Equivalence Class Analysis*, starting on page 106).

In Chapter 5, we applied the EGP tool to the epistemic gossip protocols described in Chapter 3. We analysed the performance of our epistemic gossip protocols in terms of their time and space efficiency (given respectively by the average execution length and standard extension size of these protocols). We also compared the protocols with respect to their scalability and adaptability properties. While the scalability property measures the performance of the protocols with increase in the size of the scenario, the adaptability property measures the performance of the protocols with change in the underlying network topology. We studied the performance of our protocols on various network topologies including the complete, line, binary tree, star and circle topology networks, and compared the performance of our epistemic gossip protocols with theoretical results from the traditional literature on the gossip problem. Our experiments show that the Learn New Secrets protocol has the best performance among our epistemic gossip protocols, in a complete topology network. For the tree topology networks (that is, the line, binary tree and star topology networks) and complete topology networks, Known Information Growth de Re shows a better performance than Known Information Growth de Dicto. This also shows that Known Information Growth de Re has a greater adaptability than the other protocols studied. For the scalability property, our experiments show that Learn New Secrets protocol is more scalable than the other studied protocols on a complete topology network. From our experimental results it is still unclear which of Known information Growth de Dicto and Known Information Growth de Re shows more scalability on the complete topology and binary tree topology networks. However, from the experiments we show that on the star topology network, Known Information Growth de Re is more scalable than Known Information Growth de Dicto.

In Chapter 6, we studied the theoretical properties of Learn New Secrets, Known Information Growth de Dicto, Known Information Growth de Re, Possible Information Growth de Dicto and Possible Information Growth de Re protocols. We considered the properties of these protocols on various network topologies. We proved that both Known Information Growth de Dicto and Known Information Growth de Re are successful on any tree topology network, and in particular they are successful in the line, binary tree and star topology networks. We showed that although Learn New Secrets protocol is successful in a complete topology network, it is not successful in a tree topology network for scenarios with more than two agents, and it is not successful in circle topology network for scenarios with more than three agents. We also showed that the Possible Information Growth protocols are not successful in a circle topology network for a scenario with more than three agents. We proved that the shortest execution length for Learn New Secrets, Known Information Growth de Re and Known Information Growth de Dicto is $2n - 3$

in a tree topology network. We also proved that the length of the longest execution sequence of an epistemic gossip protocol is $n(n-1)/2$ if the protocol is regular, that is, if at least one of the calling pair of agents learns some new secret in any call allowed by the protocol.

## 7.2 Future Prospects

We would like to give four directions for further research, some of which have been discussed in the relevant chapters.

**Asynchronous mode calls**

The current implementation of the EGP tool assumes gossip scenarios in which it is common knowledge among all the agents that a call is made in each round of gossiping, inasmuch as the epistemic calling condition is satisfied for some pair of agents to call each other. However, in some settings it may be desirable to allow an agent to choose not to make a given call in a round even though the calling condition is satisfied for the call. This amounts to a *skip* action by the agent, allowed in asynchronous mode calls as described in Chapter 3, and which is synonymous with the behaviour of agents who may not communicate at a set time agreed by the group, but instead choose to act at another, perhaps individually more convenient time.

In any round of gossiping and in any situation in that round, if an agent is not involved in the call that took place at the situation, it will consider it possible that no call took place at all since there is the possibility that all the agents may choose to skip all their calls. As such, if an agent has not made any call yet, it may consider it possible that no call at all has been made. In fact, such an agent will never be sure how many calls have taken place. Furthermore, some of the agents that have made calls so far may not be sure of the number of calls that have actually taken place in the scenario. Take a simple case of a scenario with five agents $a, b, c, d, e$ and with the Learn New Secrets protocol. Consider the following execution sequence in the scenario: $ab; bc; ad; .....$ After the third round of the given execution sequence, none of the agents know exactly how many calls have been made so far — agent $a$ and agent $b$ know that at least two calls have been made, but both are unsure about a third call having been made in the execution sequence since agent $a$ considers it possible that the $bc$ call was not made, and agent $b$ considers it possible that the $ad$ call was not made. However, it is possible that agent $d$ knows that at least two calls have been made so far, from the information gathered from the secrets that agent $a$ knew at the $ad$ call. Moreover, agent $e$ believes that the number of calls so far made could be zero. In addition to the possible uncertainty about the number of calls that have taken place so far in an execution sequence, we also see that an arbitrary use of the skip action by the agents could give rise to many more possible situations from the point of view of the agents, in any given round.

With a protocol like Learn New Secrets where an agent can decide which agent to call based only on the secrets it currently knows, it may seem at first sight that the asynchronous mode calls will still allow the protocol to be successful since then an agent can still call other neighbouring agents no matter how uncertain it is about which situation is the actual situation. Furthermore, in any call that actually takes place, the calling agents could refine their belief about the possible situations because the calling pair may gain some information from each other which enables them to eliminate some absurd situations. This then may lead us to envisage that perhaps over a finite number of rounds the Known Information Growth protocols may too become successful. However, there is still the possibility that an agent persistently opts for the skip action, and so there will always be some execution sequence that goes on infinitely within the protocol's extension, and all the agents will then never learn all the secrets in the scenario for that execution sequence, thus yielding an unsuccessful epistemic gossip protocol.

It may therefore be desirable to describe epistemic gossip protocols that are successful with asynchronous mode calls.

Another interesting line of future work is to consider strategic issues. Suppose that the agents are allowed to choose from a set of protocols, or from a set of possible calls due to a protocol, can an agent ensure, for example, that it is the first to know all secrets, or, for that matter not the last?

### Shortest and Longest Execution Length Protocols

In Chapter 6, we proved that for a regular gossip protocol the maximum length of an execution sequence is $n(n-1)/2$. What if one is interested in having a regular epistemic gossip protocol that gives rise to only execution sequences of maximum length? What if the epistemic gossip protocol is not required to be regular, are there successful epistemic gossip protocols that guarantee only maximum length execution sequences? An epistemic gossip protocol that produces only maximum length execution sequences is desirable when gossiping itself is the goal, that is, the longer it takes the better. (There are strands of sociology claiming the benefits of gossiping [24], in which context long sequences may be preferred). On the other hand, a related question to the foregoing is whether there are epistemic gossip protocols that guarantee the *shortest* length successful execution sequences for gossip protocols.

### Protocols with Parallel Calls and Broadcast Calls

In this thesis, we considered the case where only one pairwise call is staged in a given round of gossiping. There are other alternatives however, namely, *parallel calls* in which more than one pairwise call can be staged simultaneously. One strategy for allowing the execution of parallel calls in an epistemic gossip protocols could be to allow each of all the agents to *try* to make a call with one of the agents with whom the calling condition is satisfied for the given protocol and for the given network topology. Another strategy for making parallel calls could be through $k$-party calls, in which an agent makes a *conference*

*call* with all the agents with whom it can call in a given situation. For example, if the calling condition is satisfied for an agent $a$ to call agent $b$ and agent $c$, then we can set up a conference call among agents $a, b$ and $c$ in which all three agents exchange their secrets among themselves.

Furthermore, in this thesis we assumed that in each pairwise call the calling pair exchange all the secrets they know. This assumption can be relaxed so that one can also consider *one-way calls*, that is, calls in which only one of the callers sends its secrets to the other calling partner. This case is analogous to *text messaging* or *electronic mail* between the communicating pair. And to take this line still further, one can consider a case where one of the agents broadcasts its secrets to a group of agents rather than to one other agent, similar to the popular manner in which information in shared in social networks among a group of friends. Calls such as one-way and broadcast calls can be used to describe the spread of a disease, a news item, or a commercial within a population.

Epistemic gossip protocols that are based on parallel calls, one-way calls, broadcast calls, or any combination of various such types of calls could also be described and analysed.

### Network Topologies

In Chapter 6, we showed that on some network topologies, the Learn New Secrets protocol, the Known Information Growth protocols and the Possible Information Growth protocols are not successful for all values of $n$, where $n$ is the number of agents in a gossip scenario. Particularly, recall that none of these protocols is successful on a circle topology network for all values of $n$. An interesting research question is then: what is the minimum number of edges that can be added to an incomplete topology network such that a given protocol becomes successful for any given number of agents? The grid network is yet another interesting network topology whose theory is yet to be explored within the framework of epistemic gossip protocols, although experimental investigation of such and other networks topologies has now been enabled by the EGP tool.

# Appendix A

# BNF Grammar of EGPL

```
specification ::= BEGIN protocol_spec END;

protocol_spec ::= call_condition |  call_condition  topology_def |
                  call_condition  topology_def  equivalence_notion |
                  call_condition  equivalence_notion;

call_condition ::= LET AGENT_IDENTIFIER CALL AGENT_IDENTIFIER
                   IF LBRACE condition_expr SEMI RBRACE;

condition_expr ::= DISJUNCT agent_identifier_list COLON
                           LBRACE condition_expr_exp RBRACE |
                   CONJUNCT agent_identifier_list COLON
                   LBRACE condition_expr_exp RBRACE |
                   AGENT_IDENTIFIER KNOWS condition_expr |
                   condition_expr OR condition_expr |
                   condition_expr AND condition_expr |
                   condition_expr IMPLIES condition_expr |
                   NOT condition_expr |  set_boolean_expr |
                   LPAREN condition_expr RPAREN;

agent_identifier_list ::= agent_identifier_list
                          COMMA AGENT_IDENTIFIER | AGENT_IDENTIFIER;

condition_expr_exp ::= AGENT_IDENTIFIER KNOWS condition_expr_exp |
                       NOT condition_expr_exp | set_boolean_expr |
                       LPAREN condition_expr_exp RPAREN;

set_boolean_expr ::= set_boolean_expr AND set_boolean_expr |
                     set_boolean_expr OR set_boolean_expr |
                     set_boolean_expr IMPLIES set_boolean_expr |
                     singleton SETELEMENT set_expr |
                     singleton NOTSETELEMENT set_expr |
                     set_expr SUBSET set_expr |
                     set_expr PROPERSUBSET set_expr |
                     set_expr DOUBLEEQUAL set_expr |
                     set_expr NOTEQUAL set_expr |
```

```
37                      int_expr GREATERTHAN int_expr |
38                      int_expr GREATERTHANEQUAL int_expr |
39                      int_expr LESSTHAN int_expr |
40                      int_expr LESSTHANEQUAL int_expr |
41                      int_expr DOUBLEEQUAL int_expr |
42                      int_expr NOTEQUAL int_expr |
43                      TRUE  | FALSE |
44                      LPAREN set_boolean_expr RPAREN;
45
46 set_expr ::= set_expr MINUS set_expr |  set_expr INTERSECTION set_expr
       |
47              set_expr UNION set_expr | set_expr COMPLEMENT set_expr |
48              set_proper | singleton | empty_set | LPAREN set_expr
       RPAREN;
49
50 set_proper ::= SECRET LPAREN AGENT_IDENTIFIER RPAREN |
51                FIN LPAREN AGENT_IDENTIFIER RPAREN  |
52                LPAREN set_proper RPAREN;
53
54 singleton ::= INIT LPAREN AGENT_IDENTIFIER RPAREN |
55               LPAREN singleton RPAREN;
56
57 empty_set ::= EMPTYSET | LPAREN empty_set RPAREN;
58
59 int_expr ::= int_expr MINUS int_expr | int_expr PLUS int_expr |
60              int_expr MULTIPLY int_expr | int_expr MODULUS int_expr |
61              set_magnitude | INTEGER | LPAREN int_expr RPAREN;
62
63 set_magnitude ::= PIPE set_expr PIPE | LPAREN set_magnitude RPAREN;
64
65 topology_def ::=  TOPOLOGY LBRACE neighbourhood_list RBRACE;
66
67 neighbourhood_list  ::= neighbourhood_state |
68                         neighbourhood_state neighbourhood_list;
69
70 neighbourhood_state ::= AGENT_IDENTIFIER NEIGHBOUR AGENT_IDENTIFIER
       SEMI;
71
72 equivalence_notion ::= EQUIV_NOTION ASSIGNMENT INTEGER SEMI;
```

TABLE A.1: **EGPL BNF terminal symbols and string equivalents.**

| Symbol | String Equivalent |
|---|---|
| AGENT_IDENTIFIER | [A-Za-z][A-Za-z0-9]* |
| AND | "&&" |
| ASSIGNMENT | "=" |
| BEGIN | "begin" |
| CALL | "call" |
| COLON | ":" |
| COMMA | "," |
| COMPLEMENT | "\complement" |
| CONJUNCT | "conjunct" |
| DISJUNCT | "disjunct" |
| DOUBLEEQUAL | "==" |
| EMPTYSET | "empty" |
| END | "end" |
| EQUIV_NOTION | "equivalence_notion" |
| FIN | "fin" |
| GREATERTHAN | ">" |
| GREATERTHANEQUAL | ">=" |
| IF | "if" |
| IMPLIES | "->" |
| INIT | "init" |
| INTEGER | [1-9][0-9]* \| [0-9] |
| INTERSECTION | "\cap" |
| KNOWS | "knows" |
| LBRACE | "{" |
| LESSTHAN | "<" |
| LESSTHANEQUAL | "<=" |
| LET | "let" |
| LPAREN | "(" |
| MINUS | "-" |
| MODULUS | "%" |
| MULTIPLY | "*" |
| NEIGHBOUR | "neighbour" |
| NOT | "\neg" |
| NOTEQUAL | "!=" |
| NOTSETELEMENT | "\notin" |
| OR | "\|\|" |
| PIPE | "\|" |
| PLUS | "+" |
| PROPERSUBSET | "\subset" |
| RBRACE | "}" |
| RPAREN | ")" |
| SECRET | "secret" |
| SEMI | ";" |
| SETELEMENT | "\in" |
| SUBSET | "\subseteq" |
| TOPOLOGY | "topology" |
| UNION | "\cup" |

TABLE A.2: **EGPL set operators.**

| Operator | Meaning | Formal Symbol | Returns |
|---|---|---|---|
| \in | set membership | $\in$ | True/False |
| `\notin` | set non-membership | $\notin$ | True/False |
| `\cup` | set union | $\cup$ | Set |
| `\cap` | set intersection | $\cap$ | Set |
| − | set difference | $\setminus$ | Set |
| `\complement` | set complement | | Set |
| `empty` | empty set | $\emptyset$ | Set |
| `fin` | universal set | $\mathcal{U}$ | Set |
| `\subseteq` | secret(a) is subset of secret(b) if agent $a$ knows at most the same secrets as agent $b$ | $\subseteq$ | True/False |
| `\subset` | secret(a) is proper subset of secret(b) if agent $b$ knows every secret agent $a$ currently knows, but agent $a$ does not know every secret that agent $b$ currently knows | $\subset$ | True/False |
| `\|Y\|` | set magnitude of a set Y is the number of secrets contained in the set Y | `\|Y\|` | Natural number |

TABLE A.3: **EGPL set comparison operators.**

| **Operator** | **Meaning** | **Returns** |
|---|---|---|
| $>$ | secret(a) is greater than secret(b) if agent $a$ knows more secrets than $b$ knows | True/False |
| $<$ | secret(a) is less than secret(b) if agent $a$ knows less secrets than $b$ knows | True/False |
| $>=$ | secret(a) is greater than or equal to secret(b) if agent $a$ knows more or same number of secrets than $b$ knows | True/False |
| $<=$ | secret(a) is less than or equal to secret(b) if agent $a$ currently knows less or same number of secrets than $b$ knows | True/False |
| $==$ | secret(a) is equal to secret(b) if agent $a$ knows same secrets than $b$ knows | True/False |
| $!=$ | secret(a) is notequal to secret(b) if agent $a$ does not know the same secrets as agent $b$ | True/False |

TABLE A.4: **EGPL arithmetic and boolean operators.**

| **Operator** | **Function** | **Returns** |
|---|---|---|
| && | Conjunction | True/False |
| \|\| | Disjunction | True/False |
| `\neg` | Negation | True/False |
| $->$ | Implication | True/False |
| $-$ | Subtraction | Natural Number |
| $+$ | Addition | Natural Number |
| $*$ | Multiplication | Natural Number |
| % | Modulus | Natural Number |

# Appendix B

# More Empirical Results

## (Using Equivalence Notion 1: Definition 4.7)

TABLE B.1: **Protocol 2 on Line Topology Network.**

| Execution Sequence Length | Three Agents | Four Agents | Five Agents | Six Agents |
|---|---|---|---|---|
| 3 | 16 | | | |
| 4 | | 0 | | |
| 5 | | 192 | | |
| 6 | | 512 | 0 | |
| 7 | | | 2,048 | |
| 8 | | | 16,512 | 0 |
| 9 | | | 39,424 | 10,240 |
| 10 | | | 59,392 | 188,160 |
| 11 | | | | 1,383,168 |
| 12 | | | | 4,768,000 |
| 13 | | | | 14,789,376 |
| 14 | | | | 23,527,040 |
| 15 | | | | 31,064,576 |
| Standard Extension Size | 16 | 704 | 117,376 | 75,730,560 |
| Average Execution Length | 3 | 5.72727 | 9.33043 | 14.02358 |
| Successful Sequences | 16 | 704 | 117,376 | 75,730,560 |
| % Successful Sequences | 100.00% | 100.00% | 100.00% | 100% |

TABLE B.2: **Protocol 3 on Line Topology Network.**

| Execution Sequence Length | Three Agents | Four Agents | Five Agents | Six Agents |
|---|---|---|---|---|
| 3 | 16 | | | |
| 4 | | 0 | | |
| 5 | | 192 | | |
| 6 | | 512 | 0 | |
| 7 | | | 2,048 | |
| 8 | | | 16,512 | 0 |
| 9 | | | 39,424 | 10,240 |
| 10 | | | 59,392 | 188,160 |
| 11 | | | | 1,383,168 |
| 12 | | | | 4,768,000 |
| 13 | | | | 14,789,376 |
| 14 | | | | 23,527,040 |
| 15 | | | | 31,064,576 |
| Standard Extension Size | 16 | 704 | 117,376 | 75,730,560 |
| Average Execution Length | 3 | 5.72727 | 9.33043 | 14.02358 |
| Successful Sequences | 16 | 704 | 117,376 | 75,730,560 |
| % Successful Sequences | 100.00% | 100.00% | 100.00% | 100.00% |

TABLE B.3: **Protocol 2 on Star Topology Network.**

| Execution Sequence Length | Three Agents | Four Agents | Five Agents | Six Agents |
|---|---|---|---|---|
| 3 | 16 | | | |
| 4 | | 0 | | |
| 5 | | 288 | | |
| 6 | | 768 | 0 | |
| 7 | | | 12,288 | |
| 8 | | | 61,440 | 0 |
| 9 | | | 196,608 | 798,720 |
| 10 | | | 294,912 | 8,355,840 |
| 11 | | | | 43,253,760 |
| 12 | | | | 161,955,840 |
| 13 | | | | 471,859,200 |
| 14 | | | | 1,014,497,280 |
| 15 | | | | 1,132,462,080 |
| Standard Extension Size | 16 | 1056 | 565,248 | 2,833,182,720 |
| Average Execution Length | 3 | 5.72727 | 9.36957 | 14.05983 |
| Successful Sequences | 16 | 1,056 | 565,248 | 2,833,182,720 |
| % Successful Sequences | 100.00% | 100.00% | 100.00% | 100.00% |

TABLE B.4: **Protocol 3 on Star Topology Network.**

| Execution Sequence Length | Three Agents | Four Agents | Five Agents | Six Agents |
|---|---|---|---|---|
| 3 | 16 | | | |
| 4 | | 0 | | |
| 5 | | 288 | | |
| 6 | | 384 | 0 | |
| 7 | | | 6,144 | |
| 8 | | | 29,184 | 0 |
| 9 | | | 58,368 | 138,240 |
| 10 | | | 36,864 | 1,497,600 |
| 11 | | | | 7,971,840 |
| 12 | | | | 24,299,520 |
| 13 | | | | 42,823,680 |
| 14 | | | | 42,762,240 |
| 15 | | | | 17,694,720 |
| Standard Extension Size | 16 | 672 | 130,560 | 137,187,840 |
| Average Execution Length | 3 | 5.57143 | 8.96471 | 13.23955 |
| Successful Sequences | 16 | 672 | 130,560 | 137,187,840 |
| % Successful Sequences | 100.00% | 100.00% | 100.00% | 100.00% |

TABLE B.5: **Protocol 2 on Binary Tree Topology Network.**

| Execution Sequence Length | Three Agents | Four Agents | Five Agents | Six Agents |
|---|---|---|---|---|
| 3 | 16 | | | |
| 4 | | 0 | | |
| 5 | | 192 | | |
| 6 | | 512 | 0 | |
| 7 | | | 4,992 | |
| 8 | | | 30,080 | 0 |
| 9 | | | 75,904 | 55,552 |
| 10 | | | 100,864 | 637,952 |
| 11 | | | | 3,944,064 |
| 12 | | | | 11,803,776 |
| 13 | | | | 27,829,248 |
| 14 | | | | 41,918,592 |
| 15 | | | | 43,758,976 |
| Standard Extension Size | 16 | 704 | 211,840 | 129,948,160 |
| Average Execution Length | 3 | 5.72727 | 9.28701 | 13.82809 |
| Successful Sequences | 16 | 704 | 211,840 | 129,948,160 |
| % Successful Sequences | 100.00% | 100.00% | 100.00% | 100.00% |

TABLE B.6: **Protocol 3 on Binary Tree Topology Network.**

| Execution Sequence Length | Three Agents | Four Agents | Five Agents | Six Agents |
|---|---|---|---|---|
| 3 | 16 | | | |
| 4 | | 0 | | |
| 5 | | 192 | | |
| 6 | | 512 | 0 | |
| 7 | | | 4,992 | |
| 8 | | | 20,480 | 0 |
| 9 | | | 51,456 | 48,000 |
| 10 | | | 48,896 | 559,104 |
| 11 | | | | 3,130,432 |
| 12 | | | | 9,206,272 |
| 13 | | | | 20,747,520 |
| 14 | | | | 29,820,928 |
| 15 | | | | 28,959,616 |
| Standard Extension Size | 16 | 704 | 125,824 | 92,471,872 |
| Average Execution Length | 3 | 5.72727 | 9.14649 | 13.76135 |
| Successful Sequences | 16 | 704 | 125,824 | 92,471,872 |
| % Successful Sequences | 100.00% | 100.00% | 100.00% | 100.00% |

TABLE B.7: **Protocol 2 on Circle Topology Network.**

| Execution Sequence Length | Three Agents | Four Agents | Five Agents | Six Agents |
|---|---|---|---|---|
| 3 | 96 | | | |
| 4 | | 128 | | |
| 5 | | 1,920 | | |
| 6 | | 5,248 | 0 | |
| 7 | | | 53,440 | |
| 8 | | | 248,320 | 0 |
| 9 | | | 432,480 | 494,592 |
| 10 | | | 380,960 | 10,203,648 |
| 11 | | | | 47,009,664 |
| 12 | | | | 94,863,360 |
| 13 | | | | 127,751,040 |
| 14 | | | | 119,482,752 |
| 15 | | | | 85,650,816 |
| Standard Extension Size | 96 | 7,296 | 1,115,240 | 485,570,316 |
| Average Execution Length | 3 | 5.70175 | - | - |
| Successful Sequences | 96 | 7,296 | 1,115,200 | 485,455,872 |
| % Successful Sequences | 100.00% | 100.00% | 99.9964% | 99.98% |

TABLE B.8: **Protocol 3 on CircleTopology Network.**

| Execution Sequence Length | Three Agents | Four Agents | Five Agents | Six Agents |
|---|---|---|---|---|
| 3 | 96 | | | |
| 4 | | 128 | | |
| 5 | | 1,920 | | |
| 6 | | 5,248 | 0 | |
| 7 | | | 39,360 | |
| 8 | | | 188,960 | 0 |
| 9 | | | 319,040 | 99,840 |
| 10 | | | 280,480 | 2,665,728 |
| 11 | | | | 12,872,832 |
| 12 | | | | 26,996,736 |
| 13 | | | | 34,933,248 |
| 14 | | | | 32,731,392 |
| 15 | | | | 23,745,408 |
| Standard Extension Size | 96 | 7,296 | 827,940 | 134,115,468 |
| Average Execution Length | 3 | 5.70175 | - | - |
| Successful Sequences | 96 | 7,296 | 827,840 | 134,045,184 |
| % Successful Sequences | 100.00% | 100.00% | 99.99% | 99.95% |

# Appendix C

# More Empirical Results

## (Using Equivalence Notion 2: Definition 4.40)

TABLE C.1: **Protocol 2 on Line Topology Network.**

| Execution Sequence Length | Three Agents | Four Agents | Five Agents | Six Agents |
|---|---|---|---|---|
| 3 | 16 | | | |
| 4 | | 0 | | |
| 5 | | 192 | | |
| 6 | | 512 | 0 | |
| 7 | | | 2,048 | |
| 8 | | | 16,512 | 0 |
| 9 | | | 39,424 | 10,240 |
| 10 | | | 59,392 | 188,160 |
| 11 | | | | 1,383,168 |
| 12 | | | | 4,768,000 |
| 13 | | | | 14,789,376 |
| 14 | | | | 23,527,040 |
| 15 | | | | 31,064,576 |
| Standard Extension Size | 16 | 704 | 117,376 | 75,730,560 |
| Average Execution Length | 3 | 5.72727 | 9.33043 | 14.02358 |
| Successful Sequences | 16 | 704 | 117,376 | 75,730,560 |
| % Successful Sequences | 100.00% | 100.00% | 100.00% | 100% |

TABLE C.2: **Protocol 3 on Line Topology Network.**

| Execution Sequence Length | Three Agents | Four Agents | Five Agents | Six Agents |
|---|---|---|---|---|
| 3 | 16 | | | |
| 4 | | 0 | | |
| 5 | | 192 | | |
| 6 | | 512 | 0 | |
| 7 | | | 2,048 | |
| 8 | | | 16,512 | 0 |
| 9 | | | 39,424 | 10,240 |
| 10 | | | 59,392 | 188,160 |
| 11 | | | | 1,383,168 |
| 12 | | | | 4,768,000 |
| 13 | | | | 14,789,376 |
| 14 | | | | 23,527,040 |
| 15 | | | | 31,064,576 |
| Standard Extension Size | 16 | 704 | 117,376 | 75,730,560 |
| Average Execution Length | 3 | 5.72727 | 9.33043 | 14.02358 |
| Successful Sequences | 16 | 704 | 117,376 | 75,730,560 |
| % Successful Sequences | 100.00% | 100.00% | 100.00% | 100.00% |

TABLE C.3: **Protocol 2 on Star Topology Network.**

| Execution Sequence Length | Three Agents | Four Agents | Five Agents | Six Agents |
|---|---|---|---|---|
| 3 | 16 | | | |
| 4 | | 0 | | |
| 5 | | 288 | | |
| 6 | | 768 | 0 | |
| 7 | | | 12,288 | |
| 8 | | | 61,440 | 0 |
| 9 | | | 196,608 | 798,720 |
| 10 | | | 294,912 | 8,355,840 |
| 11 | | | | 43,253,760 |
| 12 | | | | 161,955,840 |
| 13 | | | | 471,859,200 |
| 14 | | | | 1,014,497,280 |
| 15 | | | | 1,132,462,080 |
| Standard Extension Size | 16 | 1,056 | 565,248 | 2,833,182,720 |
| Average Execution Length | 3 | 5.72727 | 9.36957 | 14.05983 |
| Successful Sequences | 16 | 1,056 | 565,248 | 2,833,182,720 |
| % Successful Sequences | 100.00% | 100.00% | 100.00% | 100.00% |

TABLE C.4: **Protocol 3 on Star Topology Network.**

| Execution Sequence Length | Three Agents | Four Agents | Five Agents | Six Agents |
|---|---|---|---|---|
| 3 | 16 | | | |
| 4 | | 0 | | |
| 5 | | 288 | | |
| 6 | | 384 | 0 | |
| 7 | | | 6,144 | |
| 8 | | | 29,184 | 0 |
| 9 | | | 58,368 | 138,240 |
| 10 | | | 36,864 | 1,497,600 |
| 11 | | | | 7,971,840 |
| 12 | | | | 24,299,520 |
| 13 | | | | 42,823,680 |
| 14 | | | | 42,762,240 |
| 15 | | | | 17,694,720 |
| Standard Extension Size | 16 | 672 | 130,560 | 137,187,840 |
| Average Execution Length | 3 | 5.57143 | 8.96471 | 13.23955 |
| Successful Sequences | 16 | 672 | 130,560 | 137,187,840 |
| % Successful Sequences | 100.00% | 100.00% | 100.00% | 100.00% |

TABLE C.5: **Protocol 2 on Binary Tree Topology Network.**

| Execution Sequence Length | Three Agents | Four Agents | Five Agents | Six Agents |
|---|---|---|---|---|
| 3 | 16 | | | |
| 4 | | 0 | | |
| 5 | | 192 | | |
| 6 | | 512 | 0 | |
| 7 | | | 4,992 | |
| 8 | | | 30,080 | 0 |
| 9 | | | 75,904 | 55,552 |
| 10 | | | 100,864 | 637,952 |
| 11 | | | | 3,944,064 |
| 12 | | | | 11,803,776 |
| 13 | | | | 27,829,248 |
| 14 | | | | 41,918,592 |
| 15 | | | | 43,758,976 |
| Standard Extension Size | 16 | 704 | 211,840 | 129,948,160 |
| Average Execution Length | 3 | 5.72727 | 9.28701 | 13.82809 |
| Successful Sequences | 16 | 704 | 211,840 | 129,948,160 |
| % Successful Sequences | 100.00% | 100.00% | 100.00% | 100.00% |

TABLE C.6: **Protocol 3 on Binary Tree Topology Network.**

| Execution Sequence Length | Three Agents | Four Agents | Five Agents | Six Agents |
|---|---|---|---|---|
| 3 | 16 | | | |
| 4 | | 0 | | |
| 5 | | 192 | | |
| 6 | | 512 | 0 | |
| 7 | | | 4,992 | |
| 8 | | | 20,480 | 0 |
| 9 | | | 51,456 | 48,000 |
| 10 | | | 48,896 | 559,104 |
| 11 | | | | 3,130,432 |
| 12 | | | | 9,206,272 |
| 13 | | | | 20,747,520 |
| 14 | | | | 29,820,928 |
| 15 | | | | 28,959,616 |
| Standard Extension Size | 16 | 704 | 125,824 | 92,471,872 |
| Average Execution Length | 3 | 5.72727 | 9.14649 | 13.76135 |
| Successful Sequences | 16 | 704 | 125,824 | 92,471,872 |
| % Successful Sequences | 100.00% | 100.00% | 100.00% | 100.00% |

TABLE C.7: **Protocol 2 on Circle Topology Network.**

| Execution Sequence Length | Three Agents | Four Agents | Five Agents | Six Agents |
|---|---|---|---|---|
| 3 | 96 | | | |
| 4 | | 128 | | |
| 5 | | 2,048 | | |
| 6 | | 5,504 | 0 | |
| 7 | | | 55,680 | |
| 8 | | | 274,560 | 0 |
| 9 | | | 514,080 | 524,544 |
| 10 | | | 464,320 | 10,930,944 |
| 11 | | | | 51,730,560 |
| 12 | | | | 108,080,640 |
| 13 | | | | 158,355,840 |
| 14 | | | | 156,101,376 |
| 15 | | | | 95,937,408 |
| Standard Extension Size | 96 | 7,680 | 1,308,680 | 581,771,892 |
| Average Execution Length | 3 | 5.70000 | - | - |
| Successful Sequences | 96 | 7,680 | 1,308,640 | 581,661,312 |
| % Successful Sequences | 100.00% | 100.00% | 99.9969% | 99.98% |

TABLE C.8: **Protocol 3 on CircleTopology Network.**

| Execution Sequence Length | Three Agents | Four Agents | Five Agents | Six Agents |
|---|---|---|---|---|
| 3 | 96 | | | |
| 4 | | 128 | | |
| 5 | | 2,048 | | |
| 6 | | 5,504 | 0 | |
| 7 | | | 41,280 | |
| 8 | | | 209,760 | 0 |
| 9 | | | 385,280 | 116,736 |
| 10 | | | 365,920 | 2,987,520 |
| 11 | | | | 14,377,728 |
| 12 | | | | 30,780,288 |
| 13 | | | | 41,245,440 |
| 14 | | | | 39,020,160 |
| 15 | | | | 25,069,440 |
| Standard Extension Size | 96 | 7,680 | 1,002,290 | 153,665,580 |
| Average Execution Length | 3 | 5.70000 | - | - |
| Successful Sequences | 96 | 7,680 | 1,002,240 | 153,597,312 |
| % Successful Sequences | 100.00% | 100.00% | 99.9950% | 99.9556% |

# Bibliography


[1] T. Ågotnes and H. van Ditmarsch. What will they say? - Public announcement games. *Synthese*, 179(S.1):57–85, 2011.

[2] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49:672–713, 2002.

[3] M. Attamah, H. van Ditmarsch, D. Grossi, and W. van der Hoek. Knowledge and gossip. In *Proceedings of ECAI 2014*, volume 263 of *Frontiers in Artificial Intelligence and Applications*, pages 21 – 26. IOS Press, 2014.

[4] M. Attamah, H. van Ditmarsch, D. Grossi, and W. van der Hoek. A framework for epistemic gossip protocols. In N. Bulling, editor, *Multi-Agent Systems*, volume 8953 of *Lecture Notes in Artificial Intelligence*, pages 193–209. Springer, 2015.

[5] M. Attamah, H. van Ditmarsch, D. Grossi, and W. van der Hoek. The pleasure of gossip. In *Rohit Parikh on logic, language and society*. Springer, 2015.

[6] G. Aucher. DEL-sequents for progression. *Journal of Applied Non-Classical Logics*, 21(3-4):289–321, 2011.

[7] G. Aucher. DEL-sequents for regression and epistemic planning. *Journal of Applied Non-Classical Logics*, 22(4):337–367, 2012.

[8] A. Bavelas. Communication patterns in task-oriented groups. *The Journal of the Acoustical Society of America*, pages 725–730, 1950.

[9] P. Blackburn, M. De Rijke, and Y. Venema. *Modal logic*, volume 53. Cambridge University Press, 2002.

[10] T. Bolander and M. Andersen. Epistemic planning for single and multi-agent systems. *Journal of Applied Non-classical Logics*, 21(1):9–34, 2011.

[11] D. Boyd and J. Steele. Random exchanges of information. *Journal of Applied Probability*, pages 657–661, 1979.

[12] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Randomized gossip algorithms. *IEEE/ACM Trans. Netw.*, 14(SI):2508–2530, 2006.

[13] R. Bumby. A problem with telephones. *SIAM Journal on Algebraic Discrete Methods*, 2(1):13–18, 1981.

[14] I. Cederbaum. On the spread of information in communication nets. *Matrix and Tensor*, 30:101–112, 1980.

[15] K. Chandy and J. Misra. How processes learn. In *PODC '85: Proc. of the fourth annual ACM symposium on Principles of distributed computing*, pages 204–214, New York, NY, USA, 1985. ACM.

[16] A. Church. Application of recursive arithmetic to the problem of circuit synthesis. In *Summaries of the Summer Institute of Symbolic Logic, vol. I*, pages 3–50. Cornell University, 1957.

[17] A. Church. Logic, arithmetic and automata. In *Proceedings of the international congress of mathematicians*, pages 23–35, 1962.

[18] N. Cot. Extensions of the telephone problem. In *Proc. 7th SE Conf. on Comb. Graph Theory and Comp., Utilitas Mathematica, Winnipeg, Manitoba, Canada*, pages 239–256, 1976.

[19] F. L. DeRemer. *Practical translators for LR (k) languages.* PhD thesis, Massachusetts Institute of Technology, 1969.

[20] E. A. Emerson and J. Y. Halpern. Decision procedures and expressiveness in the temporal logic of branching time. In *Proceedings of the fourteenth annual ACM symposium on Theory of computing*, pages 169–180. ACM, 1982.

[21] E. A. Emerson and J. Y. Halpern. "sometimes"Ăİ and "ĂIJnot never"Ăİ revisited: on branching versus linear time temporal logic. *Journal of the ACM (JACM)*, 33(1):151–178, 1986.

[22] R. Fagin, J. Halpern, Y. Moses, and M. Vardi. *Reasoning about Knowledge.* MIT Press, Cambridge MA, 1995.

[23] A. Farley and A. Proskurowski. Gossiping in grid graphs. *Journal of Combined Information System Sciences*, 5:161–172, 1980.

[24] M. Feinberg, R. Willer, J. Stellar, and D. Keltner. The virtues of gossip: Reputational information sharing as prosocial behavior. *Journal of Personality and Social Psychology*, 102:1015–1030, 2012.

[25] P. Gammie and R. van der Meyden. MCK: Model checking the logic of knowledge. In R. Alur and D. Peled, editors, *Proc. of the 16th International Conference on Computer Aided Verification (CAV 2004)*, pages 479–483. Springer, 2004.

[26] M. Golumbic. The general gossip problem. Technical Report RC 4977, IBM, August 1974. IBM research report.

[27] A. Hajnal, E. Milner, and E. Szemerédi. A cure for the telephone disease. *Canad. Math. Bull*, 15(3):447–450, 1972.

[28] F. Harary and A. Schwenk. The communication problem on graphs and digraphs. *Journal of the Franklin Institute*, 297(6):491–495, 1974.

[29] F. Harary and A. Schwenk. Efficiency of dissemination of information in one-way and two-way communication networks. *Behavioral Science*, 19(2):133–135, 1974.

[30] S. Hedetniemi, S. Hedetniemi, and A. Liestman. A survey of gossiping and broadcasting in communication networks. *Networks*, 18:319–349, 1988.

[31] S. E. Hudson, F. Flannery, C. S. Ananian, and D. Wang. Cup lalr parser generator for java. `http://www2.cs.tum.edu/projects/cup/`, 1999. [Online; accessed 24-July-2014].

[32] C. Hurkens. Spreading gossip efficiently. *Nieuw Archief voor Wiskunde*, 5/1(2):208–210, 2000.

[33] M. Huth and M. Ryan. *Logic in Computer Science: Modelling and reasoning about systems*. Cambridge University Press, 2004.

[34] W. Jamroga and W. van der Hoek. Agents that know how to play. *Fundamenta Informaticae*, 63:185–219, 2004.

[35] D. Kempe, A. Dobra, and J. Gehrke. Gossip-based computation of aggregate information. In *Proceedings of the 44th FOCS*, pages 482–491. IEEE Computer Society, 2003.

[36] D. Kleitman and J. Shearer. Further gossip problems. *Discrete Mathematics*, 30(2):151–156, 1980.

[37] W. Knödel. New gossips and telephones. *Discrete Mathematics*, 13:95, 1975.

[38] D. E. Knuth. On the translation of languages from left to right. *Information and control*, 8(6):607–639, 1965.

[39] O. Kupferman and M. Vardi. Church's problem revisited. *Bull. Symbolic Logic*, 5(2):245–263, 1999.

[40] L. Lamport. "sometimes" is sometimes "not never": On the temporal logic of programs. In *Proceedings of the 7th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 174–185. ACM, 1980.

[41] H. Landau. The distribution of completion times for random communication in a task-oriented group. *Bull. Math. Biophys.*, 16(3):187–201, 1954.

[42] A. Lomuscio, H. Qu, and F. Raimondi. MCMAS: A model checker for the verification of multi-agent systems. In *Computer Aided Verification*, pages 682–688. Springer, 2009.

[43] A. Lomuscio and M. Sergot. Deontic interpreted systems. *Studia Logica*, 75(1):63–92, 2003.

[44] B. Löwe, E. Pacuit, and A. Witzel. Del planning and some tractable cases. In *Logic, Rationality, and Interaction*, pages 179–192. Springer, 2011.

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

[46] D. Miklos, M. Newman, A. Seress, and D. West. The addition game: a variant of gossiping without duplication. *Discrete Mathematics*, 68:265–272, 1988.

[47] J. Moon. Random exchanges of information. *Nieuw Arch. Wisk*, 20:246–249, 1972.

[48] L. Panait and S. Luke. Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems*, 11(3):387–434, 2005.

[49] R. Parikh and P. Krasucki. Levels of knowledge in distributed systems. *Sadhana*, 17(1):167–191, 1992.

[50] 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.

[51] R. Parikh and R. Ramanujam. A knowledge based semantics of messages. *Journal of Logic, Language and Information*, 12:453–467, 2003.

[52] W. Penczek and A. Lomuscio. Verifying epistemic properties of multi-agent systems via bounded model checking. In *Proc. of 2nd AAMAS*, pages 209–216. ACM, 2003.

[53] A. Pnueli and R. Rosner. On the synthesis of an asynchronous reactive module. In *ICALP '89: Proc. of the 16th International Colloquium on Automata, Languages and Programming*, pages 652–671. Springer, 1989.

[54] A. Pnueli and R. Rosner. Distributed reactive systems are hard to synthesize. In *SFCS '90: Proceedings of the 31st Annual Symposium on Foundations of Computer Science (vol.2)*, pages 746–757. IEEE Computer Society, 1990.

[55] J. Ruan. *Reasoning about Time, Action and Knowledge in Multi-Agent Systems*. PhD thesis, University of Liverpool, 2008.

[56] M. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, New Jersey, 2010. Third Edition.

[57] Á. Seress. Gossiping old ladies. *Discrete Mathematics*, 46(1):75–81, 1983.

[58] Á. Seress. Gossips by conference calls. *Stud. Sci. Math. Hung*, 20, 1985.

[59] Á. Seress. Quick gossiping without duplicate transmissions. *Graphs and Combinatorics*, 2(1):363–381, 1986.

[60] A. Shimbel. Applications of matrix algebra to communication nets. *Bull. Math. Biol.*, 13(3):165–178, 1951.

[61] S. Shostak and B. Baker. Gossips and telephones. *Discrete Mathematics*, 2(3):191–193, 1972.

[62] S. Skiena. Implementing discrete mathematics: Combinatorics and graph theory with mathematica, 1990.

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

[64] J. van Benthem. Games in dynamic epistemic logic. *Bulletin of Economic Research*, 53(4):219–248, 2001.

[65] R. van der Meyden and M. Vardi. Synthesis from knowledge-based specifications (extended abstract). In *CONCUR '98: Proceedings of the 9th International Conference on Concurrency Theory*, pages 34–49. Springer, 1998.

[66] H. van Ditmarsch and B. Kooi. Semantic results for ontic and epistemic change. In G. Bonanno, W. van der Hoek, and M. Wooldridge, editors, *Logic and the Foundation of Game and Decision Theory (LOFT 7)*, Texts in Logic and Games 3, pages 87–117. Amsterdam University Press, 2008.

[67] H. van Ditmarsch, W. van der Hoek, and B. Kooi. *Dynamic Epistemic Logic*, volume 337 of *Synthese Library*. Springer, 2007.

[68] J. van Eijck. DEMO — a demo of epistemic modelling. In J. van Benthem, D. Gabbay, and B. Löwe, editors, *Interactive Logic — Proc. of the 7th Augustus de Morgan Workshop*, pages 305–363. Amsterdam University Press, 2007. Texts in Logic and Games 1.

[69] D. West. A class of solutions to the gossip problem, part i. *Discrete Mathematics*, 39(3):307–326, 1982.

[70] M. Wooldridge. *An introduction to multiagent systems*. John Wiley & Sons, 2009.