Thesis for the Degree of Doctor of Philosophy

Privacy Policies for Social Networks
A Formal Approach

Raúl Pardo

Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
Gothenburg, Sweden 2017

Privacy Policies for Social Networks — A Formal Approach
Raúl Pardo
ISBN 978-91-7597-631-0 (Print), 978-91-7597-631-0 (PDF)

Technical report 146D
Department of Computer Science and Engineering
Research groups: Formal Methods and Language Technology

Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Sweden
Telephone +46 (0)31-772 1000

Printed at Chalmers reproservice
Gothenburg, Sweden 2017

Abstract

Online Social Networks (OSNs) are ubiquitous, with more than 70% of Internet users being part of them. The pervasive nature of OSNs brings many threats and challenges, privacy being one of them. Very often the available privacy protection mechanisms in OSNs do not meet users requirements. This results in users that are unable to define privacy settings (also known as privacy policies) that meet their expectations. Furthermore, current privacy settings are difficult to understand, which makes users sharing their personal information with more people than they actually intend to. In this thesis we explore novel techniques to protect users' privacy in OSNs.

On the one hand, we define a formal framework to write privacy policies in OSNs and to reason about them. We use this framework to define and study current and *new types of privacy policies* that are not present in today's OSNs. In particular, we look into: *i)* protection against implicit disclosure of information, e.g., a user sharing someone else's information—without her consent; and *ii)* evolving privacy policies, i.e., privacy policies that change over time, e.g., *"my supervisor cannot see my location during the weekend"*. These formalisms also provide a direct enforcement mechanism for this new type of privacy policies. We have developed a proof-of-concept implementation of the enforcement to show the practicality of our technique. We formally prove that this enforcement is correct, i.e., no privacy violations may occur.

On the other hand, we look into the problem of *embedding privacy policies into the data.* Having policies and data as separate entities is prone to consistency issues. It might happen that the data is accessed by individuals who should not have access to it because the access policy is outdated or simply missing. This issue is particularly important in OSNs as they normally rely on geographically distributed databases or have a distributed architecture. Concretely, we use Attributed-Based Encryption (ABE) to "attach" privacy policies to pictures.

Keywords: *online social networks, privacy, formal methods, epistemic logic*

To Diego, Paqui, Carmen and Llanos.

Funding

The work presented in this thesis has been funded by the Swedish funding agency SSF under the grant *Data Driven Secure Business Intelligence*.

Acknowledgements

I would like to thank all the people who supported me during the exciting process of becoming a PhD.

Firstly, I thank my advisor Gerardo Schneider for his excellent guidance and for giving me the opportunity to work together. Thank you, when I look back, I see how much you have helped me to grow as a researcher and as a person.

Special thanks to my collaborators. It has been a pleasure working with all of you and I have learnt a lot from the experience. This thesis would not have been possible without you.

I want to thank Musard Balliu for his insights on my research and advise which are now reflected on this thesis. I am also grateful to Andrei Sabelfeld for always suggesting new ideas to explore, his comments on an earlier version of this thesis, and for the energizing Sunday runs in the forest.

Thanks to the members of the Formal Methods division, I really enjoyed our weekly seminars that have contributed to substantially expand my knowledge on Formal Methods. In particular, I would like to thank Wolfgang Ahrendt for giving me new viewpoints when presenting my research.

Devdatt Dubhashi deserves special thanks for extensive debates on Formal Methods versus Machine Learning—among others—with which I have developed excellent discussion skills.

I am very grateful to all my friends at the Department. Thank you guys for being there to hang out, go dancing, go climbing, the trips, and all experiences we have shared. You have made my stay in Sweden one of the most enjoyable moments of my life.

I would also like to thank Fernando L. Pelayo, who introduced me to the research world and he has been a constant help ever since. Thank you very much.

Quiero agradecer el apoyo incondicional de mis padres Diego y Paqui, gracias a vosotros he llegado hasta aquí. A mis hermanas Carmen y Llanos quiero agradecerles que siempre están ahí. También agradezco a mis abuelos, tíos y primos su gran apoyo.

No me olvido de amigos Rafa, Callejas, Juanma, Kike, Lucio, Andrés, Isra, Villa, Bea, Pedro, Laura, Lacoste, Ana, Juanma Soler, Rubén, Pedro García y Javi. Gracias por hacer que cada vez que os vuelvo a ver parezca que el tiempo no ha pasado.

Finally, thanks to Mahsa for making this year very special, and for your continuous support.

Structure of this thesis

This thesis comprises a collection of seven scientific articles devoted to exploring different techniques to offer advanced privacy settings in online social networks. The articles correspond to Chapters II-VIII and are published in the following venues:

Chapter II Raúl Pardo and Gerardo Schneider. 'A Formal Privacy Policy Framework for Social Networks'. In: *Proceedings of the 12th International Conference on Software Engineering and Formal Methods, SEFM'14*. Vol. 8702. LNCS. 2014, pp. 378–392. ISBN: 978-3-319-10430-0. DOI: `10.1007/978-3-319-10431-7_30`

Chapter III Raúl Pardo, Musard Balliu, and Gerardo Schneider. 'Formalising privacy policies in social networks'. In: *Journal of Logical and Algebraic Methods in Programming* (2017). ISSN: 2352-2208. DOI: `10.1016/j.jlamp.2017.02.008`

Chapter IV Raúl Pardo and Gerardo Schneider. 'Model Checking Social Network Models'. In: *Proceedings of the Eighth International Symposium on Games, Automata, Logics and Formal Verification, GandALF'17*. Vol. 256. EPTCS. 2017, pp. 238–252. DOI: `10.4204/EPTCS.256.17`

Chapter V Raúl Pardo, Christian Colombo, Gordon J. Pace, and Gerardo Schneider. 'An Automata-Based Approach to Evolving Privacy Policies for Social Networks'. In: *Proceedings of the 16th International Conference on Runtime Verification, RV'16*. Vol. 10012. LNCS. 2016, pp. 285–301. ISBN: 978-3-319-46981-2. DOI: `10.1007/978-3-319-46982-9_18`

Chapter VI Raúl Pardo, Ivana Kellyérová, César Sánchez, and Gerardo Schneider. 'Specification of Evolving Privacy Policies for Online Social Networks'. In: *Proceedings of the 23rd International Symposium on Temporal Representation and Reasoning, TIME'16*. IEEE, 2016, pp. 70–79. ISBN: 978-1-5090-3825-1. DOI: `10.1109/TIME.2016.15`

Chapter VII Raúl Pardo, César Sánchez, and Gerardo Schneider. 'Timed Epistemic Knowledge Bases for Social Networks (Extended Version)'. In: *ArXiv e-prints* (2017). eprint: `1708.04070` (cs.LO)

Chapter VIII Pablo Picazo-Sanchez, Raúl Pardo, and Gerardo Schneider. 'Secure Photo Sharing in Social Networks'. In: *Proceedings of the 32nd International Conference on ICT Systems Security and Privacy Protection, IFIP SEC 2017*. Vol. 502. 2017, pp. 79–92. ISBN: 978-3-319-58469-0. DOI: `10.1007/978-3-319-58469-0_6`

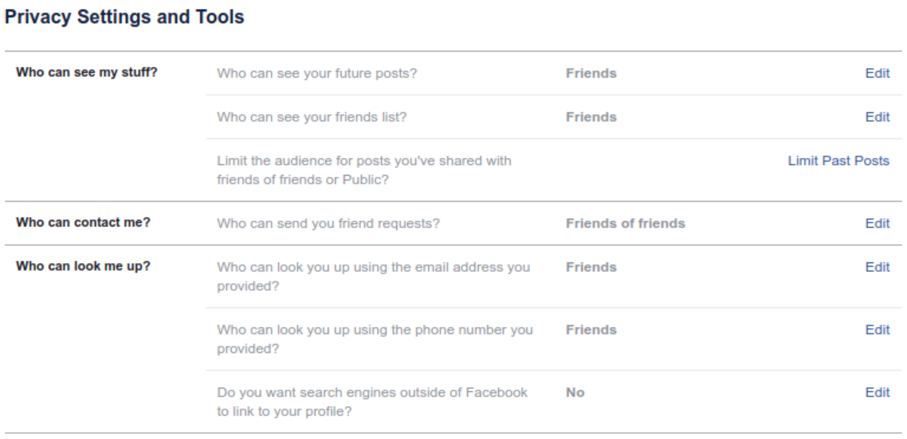
Contents

Chapter I

Introduction

As the use of *Online Social Networks* (OSNs) increases [45], privacy breaches in OSNs keep pace with this growth [81, 56, 41, 51]. This is not a surprise. The main purpose of OSNs is to share information. OSN users share mostly personal information related to the social aspects of their lives, e.g., pictures, locations, birthday, relationship status, political opinions and so forth. That is, users' sensitive information. Privacy, however, has been an afterthought, and privacy breaches are being fixed as they appear. A privacy breach occurs when user information is disclosed to an undesired audience or when the information is misused. The author of a privacy breach can be either the OSN provider or its users. An example of the former is an OSN provider that sells users data to third parties without users' consent. A user tagging another user in a picture or re-sharing a post—thus increasing the post's audience—are examples of the latter. In this thesis we focus on privacy breaches caused by users. Most of the time this type of privacy breaches is related to the lack of control that users have over the information they share, and difficulty understanding who can see their information.

In this chapter, we look into the tools that OSN users have to protect their privacy and give an overview of our proposal on how to improve them. First, we start by describing *privacy policies* for OSNs and related problems (Section I.1). Second, we look into the protection mechanism that OSN providers currently use to implement privacy settings and discuss their limitations (Section I.2). Finally, we give an overview of this thesis where we present our contributions to solve the privacy problems we describe (Section I.3).



Privacy Settings and Tools

Who can see my stuff?	Who can see your future posts?	Friends	Edit
	Who can see your friends list?	Friends	Edit
	Limit the audience for posts you've shared with friends of friends or Public?		Limit Past Posts
Who can contact me?	Who can send you friend requests?	Friends of friends	Edit
Who can look me up?	Who can look you up using the email address you provided?	Friends	Edit
	Who can look you up using the phone number you provided?	Friends	Edit
	Do you want search engines outside of Facebook to link to your profile?	No	Edit

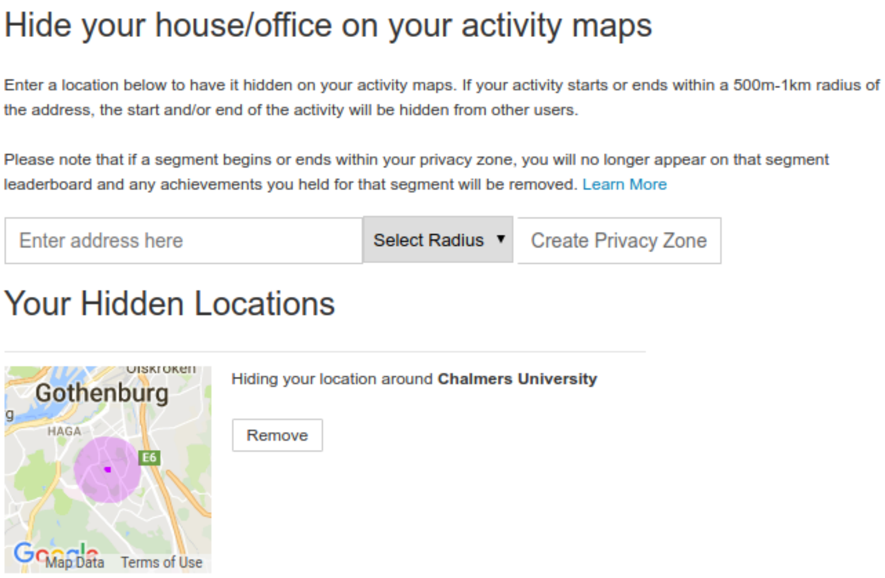
Figure I.1: Facebook Privacy Settings

I.1 Privacy Policies in Online Social Networks

Users in OSNs express their privacy preferences by means of *privacy policies*. Current privacy policies define *who* can access *what* information. For instance, in the privacy settings[1] of Facebook, users can specify who can see their posts or friends list—first row in Fig. I.1. They can also limit the actions that other users can perform, e.g., they can limit who can send them a friend request—fourth row in Fig. I.1. Other OSNs have privacy policies that are targeted to the particular information that users share. Consider the fitness OSN Strava, where users can upload running, cycling or swimming workouts. Usually, these workouts include GPS traces which show the locations where users have been training. In Strava, users can define *privacy zones* which are hidden areas of their workout map (cf. Fig. I.2). Finally, in almost all OSNs, users can *block* other users. Blocking forbids any interaction between the two users, hence avoiding any abusive or harmful behaviour towards their social status in the platform.

Unfortunately, empirical studies have shown that the privacy policies available in OSNs today do not meet users expectations [56, 41, 51, 1]. In [51], Liu *et al.* conducted a study where they found out that Facebook's privacy settings match users expectations only 37% of the time. In one of the experiments, the authors report that 63% of the pictures had privacy settings that were inconsistent with users' desired settings. The reasons for this are manifold. Sometimes users lack of the appropriate understanding

[1]Privacy settings are an instance of privacy policies. In the thesis, we talk about privacy policies because it is a broader term that captures more concepts than the concrete privacy settings that OSN providers implement today.



Hide your house/office on your activity maps

Enter a location below to have it hidden on your activity maps. If your activity starts or ends within a 500m-1km radius of the address, the start and/or end of the activity will be hidden from other users.

Please note that if a segment begins or ends within your privacy zone, you will no longer appear on that segment leaderboard and any achievements you held for that segment will be removed. Learn More

Your Hidden Locations

Hiding your location around **Chalmers University**

Figure I.2: Strava Privacy Zones

of the use of the OSN. Most OSNs companies spend a lot of resources in investigating how to present privacy settings to users in an effective way [63, 14]. Due to this lack of understanding, some users fall into using the default settings, which often are not protective enough [36]. Last but not least, developers lack tools that prevent privacy disclosures. Many times, privacy breaches appear after adding new functionality to the OSN. In this thesis we focus on following three concrete privacy problems:

Problem i) *Mismatch between privacy policy and user intentions.*

Problem ii) *Privacy concerns change over time.*

Problem iii) *Outdated privacy policies disclose information to the wrong audience.*

In what follows we give examples to illustrate how these three issues fail to meet users' expectations.

Mismatch between privacy policy and user intentions. This problem arises when the activation of a privacy policy does not result in the desired intention of the user. Consider a user that has the so called *friends only* profile [89]. It consists in setting all privacy policies to friends only. Fig. I.1 is an example of a friends only profile. All sections are set to Friends, except for "who can send me a friend request" which is set to friends of friends—this is the most restrictive option available. The goal of having a friend only profile is to limit the audience of *all* the user's information to only her friends. In fact having a friends only profile is one the so called *privacy-enhancing practices* for OSNs [89]. Since friendship in Facebook is a reciprocal relationship—i.e., both users need to agree to start being friends—this practice limits the audience of the information to a set of known people. Imagine now that Alice—who has a friends only profile—shares a picture of herself together with Bob. Later she tags Bob on the picture. This action will increase the picture's audience to Alice's friends and Bob's friends, thus mismatching Alice's intention of sharing her information only with her friends. Note that Alice has set all her privacy settings to friends only and she is the only one interacting with the picture, yet the picture is shown to more people than she expected.

Privacy concerns change over time. Depending on when or how many times some information is released users might prefer to share it or keep it private. For example, it may be safe to have your location disclosed twice a month. It is not so safe to have it disclosed every second. The latter might be considered a violation of privacy, considering that it does not allow the user to visit any place without it being recorded. Imagine that you are being tracked 24/7; would that influence your behaviour? Moreover, from such a detailed set of information, it might be possible to infer other information that the user does not want to disclose, e.g., favourite restaurants, most frequently visited people, weekend trips, and so on. This type of privacy flaw was found in Facebook's messenger app [9]. The app included, by default, the location of users in every message. This was exploited by other users who were able to track friends just by chatting with them. Facebook developers realised this issue and quickly changed the defaults and updated the app to alert users of the consequences of sharing their location in the app.

Bauer *et al.* empirically show that privacy concerns vary over time [6]. In their experiments they note that users change the audience of old posts depending on several factors. For instance, sometimes after having other users commenting in a post, the owner of the post decided to reduce the audience. This may occur due to comments that include political opinions, racism or sexism to which the owner of the post does not want to be linked. They also found out that as posts get old, many users prefer to reduce their audience. Even events that occur in the present might affect to whom the content is made visible in the future. Recently, Facebook introduced a breakup mode which limits the updates of people who were in a couple after breaking up [62]. In this mode, users may select not to share pictures with their ex-partners, but still allow them to send private messages. Also, the newsfeed includes less posts of the ex-partners, and they are not suggested as people to tag on a picture. By offering these advanced privacy settings, Facebook adapts better to the privacy concerns of the users after the breakup.

Outdated privacy policies disclose information to the wrong audience. This problem occurs when a user changes her privacy policies and these are not correctly updated—internally in the storage system of the OSN; thus disclosing information to undesired audience. All OSNs are geographically distributed systems, either because they have a distributed architecture such as Diaspora, or because they store information in a geographically distributed database, e.g., Facebook, Twitter or Instagram. Internally, policy and data are two separate entities. Suppose that Alice shares a picture together with a policy that allows only Bob and Charlie to access it. The picture and the policy are replicated in all nodes of the system. Afterwards Alice decides to remove Bob from the picture's audience. Now the new policy must be updated in all nodes. However, if the system uses an outdated version of the policy, the picture would still be shared with Bob. In Facebook, this issue is almost negligible, since they rely on a master/slave data store which updates all the information—including policies—in the database in a few milliseconds [11]. Yet Facebook mentions that they prioritise availability and performance, as opposed to data consistency, and they would benefit from reducing the amount of updates in the data store. In the case of Diaspora, when information is copied to different nodes of the system, policies are also copied. Unfortunately, when policies are updated in one node, they are not transmitted to all nodes where the data was copied, therefore, disclosing the information to the wrong audience.

Many of the privacy problems described in this section arise due to the underlying privacy protection mechanism of OSNs. In the following section we describe this protection mechanism and discuss its limitations.

I.2 Access Control in Online Social Networks

Access control is the protection mechanism that today's OSNs use to handle access to information. Here we provide background on the internal structure of OSNs and the access control model they implement.

I.2.1 Structure of Online Social Networks

According to Boyd and Ellison [10] OSNs have three distinguishing characteristics that differentiate them from other online services:

- A public or semi-public profile defined by users;

- A set of connections or relationships between users of the system;
- The ability for users to see certain information about others they are connected to, including meta-information as for instance others' connections.

The underlying structure of an OSN is the so called *social graph* [27, 11]. Nodes, in this graph, represent users and resources—e.g., posts, pictures, locations, and so on—and edges are used to model connections among users and resources. Fig. I.3 shows an example of such a graph. This social graph has three users: Alice, Bob and Charlie. It also contains one resource, a picture, denoted as Pic_1. Dotted arrows represent social connections between users. The arrow between Alice and Bob means that Alice and Bob are friends, and the arrow from Bob to Charlie shows that Bob follows Charlie. Note that *friend* is a bidirectional relation, meaning that Alice is a friend with Bob and vice versa. Friend is the main connection in Facebook. In Facebook it is also bidirectional—becoming friends results from a mutual agreement between users. Not all connections are bidirectional, a frequent connection between users is *follow*. It is present in several OSNs, e.g., Twitter, Instagram, Snapchat, Youtube and others. In our example—as in the mentioned OSNs—*follow* is unidirectional only indicating that Bob follows Charlie. Additionally, there are two connections between users and the picture Pic_1, as denoted by plain lines. The connection between Alice and Pic_1 indicates that she is the owner of the picture, and the remaining arrow models that Bob is tagged in Pic_1.

I.2.2 Relationship-Based Access Control

In *Relationship-Based Access Control* (ReBAC) users specify who can access their information by means of social relationships. This new approach to access control arises as a consequence of the structure of OSNs. In [32], Fong *et al.* identify distinctive features of ReBAC compared with other access control paradigms such as *Discretionary Access Control* (DAC) [34, 47] or *Role-Based Access Control* (RBAC) [82]:

- *Access policies are defined in terms of the relations in the social graph.* Moreover, each user defines the audience of her own resources. Consider Alice, in Fig. I.3, who may define that Pic_1—a picture she owns—can only be accessed by her friends, that is, Bob. Alternatively, she can set Pic_1 as public which results in Bob and Charlie having access to it.
- *For accessing any resource, it must be reachable in the social graph.* When Bob access Pic_1, permission is granted if Bob's node is reachable from Alice's through the *friend* relation. In Fig. I.3, Bob's node is reachable from Alice's. From Char-

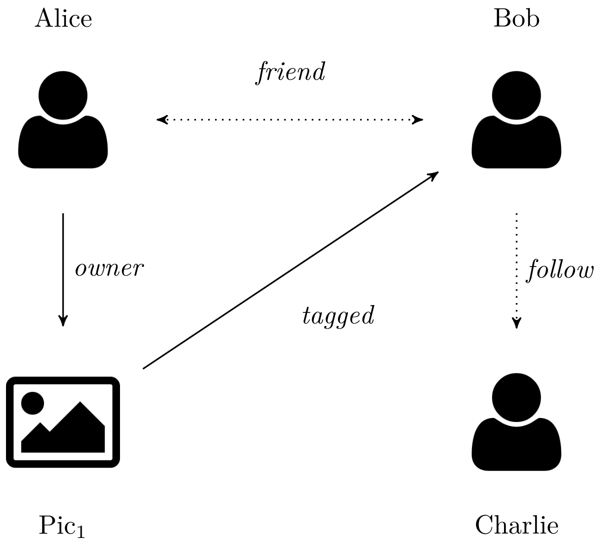


Figure I.3: Social Graph Example

lie's node, however, Bob's node is not reachable. Therefore, Charlie would not be able to access Alice's picture.

- *Abstraction of the sequence of events that occur in the system.* In many access control systems, authorisation is a function of a sequence of events [83]. Imagine that friendship in our example is a bidirectional relation which requires both users to agree before establishing the connection. In order for Charlie to access Alice's posts, one possibility is that: i) Charlie sends a friend request to Alice; ii) Alice accepts the request; iii) Charlie access the post. In ReBAC, most of the time, the sequence of events is "registered" in the social graph, hence boiling down access control to checking the social graph. For instance, in our previous example, the effect of Alice and Charlie becoming friends results in adding a *friend* relation between them. Though this approach to abstracting the event history is also present in RBAC [82], the two models are structurally different. On the one hand, in RBAC, the execution of a sequence of events results in placing users in a role or a set of roles which determine what they can access. On the other hand, in ReBAC, the execution of a sequence of events creates relations between users that condition who can access the resources in the system.

While ReBAC is tailored to the structure of OSNs, the constant presence of privacy breaches in OSN suggests that the model might need to be revised. Mondal *et al.* claim that access control is inadequate for managing privacy in OSNs [60]. They raise, among others, the following issues:

1. *Users must, a priori, specify the audience of their information.* This issue relates to *Problem ii)* mentioned above, i.e., users privacy concerns might change over time. There is a need for policies that adapt as time elapses or events occur. For instance, the approach of Dougherty *et al.* introduces an access control system with support for dynamic policies [23]. Nevertheless, it has not been studied how to adapt Dougherty's work to OSNs.

2. *Access control does not capture how information should be redistributed.* In OSNs, it is also important to let users limit the way other users can interact with their information. For example, Alice might feel comfortable sharing Pic_1 with Bob, but maybe she does not want Bob to re-share Charlie—who, initially, did not have access to Pic_1. It is important to provide users with a mechanism that helps them define how people with access to their information can use it. The recent line of work on *usage control* is tackling this issue [42]. This issue is related to *Problem i)*.

In the next Section, we introduce our approach to tackling some of the privacy issues we have described. We present new models that are aimed at granting access to information, and controlling how this information propagates in the system. These models preserve the topology of the social graph, but are enriched with additional machinery focused on enhancing the control that users have over the information.

I.3 Thesis Overview

The work included in this thesis aims at developing privacy policies for OSNs that give users more control over their information. The thesis comprises seven chapters. Chapters II-VI and VIII correspond to papers that have been published in peer-reviewed conferences, symposia and journals. Chapter VII corresponds to a paper which is currently under submission. Even though each of the papers tackles a different problem, they can be classified in the following topics: ***static privacy policies***, ***dynamic privacy policies*** and ***unification of privacy policies and data***. Fig. I.4 gives an overview of this classification. The figure also shows the relations between chapters and the problem that each topic addresses. In what follows we briefly describe each of the

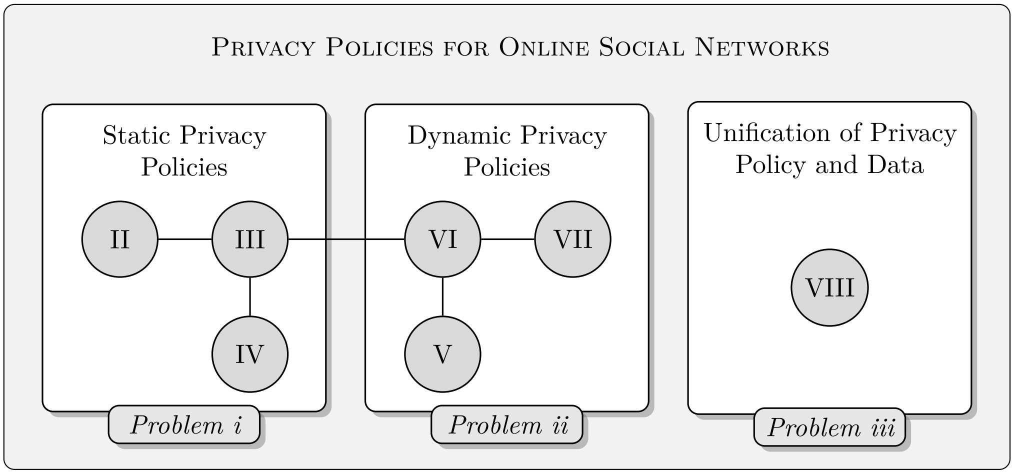


Figure I.4: Classification of the chapters in the thesis

topics covered in the thesis, how chapters relate to each other and their contribution to solving the problems mentioned in Section I.1.

Static privacy policies

In Chapters II, III and IV we present a novel framework to write privacy policies in OSNs. The framework is based on models that preserve the structure of OSN, i.e., the social graph. These models, however, are enriched with the knowledge of the users. In a nutshell, our models are social graphs enriched with a *knowledge base* for each user containing the set of facts they know. This enables users to write privacy policies in terms of the knowledge. That is, users can decide who can *know*—in other words, access—their information. For example users can specify privacy policies like *"Only my friends can know my location"*.

Although this type of policies can also be stated in ReBAC, the main difference lies in the enforcement mechanism of the framework. In ReBAC policies talk about resources, whereas in our framework policies talk about the knowledge of the users. Consider again the example we described in *Problem i)* (cf. Section I.1) where users set all privacy policies to friends only. We mentioned that, by tagging on a picture, the audience of the picture might increase hence violating the policy. In our framework, the event of tagging increases the knowledge base of the users that are part of the audience of the picture and are not friends of the owner. As a result, the enforcement mechanism detects that there are users, who are not friends with the owner of the picture, that "know" (have accessed) the picture. At this point any preventive action can be taken, for instance, blocking the tagging or hiding the picture to the users who are not friends with the owner of the picture.

Our framework is formal. We use a knowledge-based language—similar to that of epistemic logic [29]—to describe and reason about the knowledge of the users. We use a labelled transition system to formalise the evolution of the system. Small step operational semantics describes changes in the knowledge of the users, topology of the social graph and privacy policies. The formalisation gives the required tools to implement an enforcement for the privacy policies in the framework. Furthermore, using all these elements we can formally prove that no privacy policy can be violated in the system.

Dynamic privacy policies

In Chapters VI and VII we extend the framework to support privacy policies that depend on time and events. For example, the framework allows users to write a privacy policy states *"My location may be disclosed only three times per day"*. This policy depends on events (disclosing the user location) and time (every day). As opposed to static policies, this type of privacy policies is not present in any OSN today. Using dynamic privacy policies relates to *Problem ii)* presented in Section I.1, and can be used to address some of the concerns raised by Bauer *et al.* in [6]. For instance, users could specify a policy saying that *"whenever someone makes a sexist comment in one of my post, reduce the audience of the post to nobody"*, or *"reduce the audience of posts that have been uploaded for more than two months to family only"*.

On the one hand, we focus on specifying dynamic privacy policies. Chapter VI presents a temporal logic with the operators *always* and *eventually*, which allows users to specify that certain static policy must always hold in a certain interval of time. Chapter VII introduces a more expressive logic where the knowledge operator includes a time-stamp, thus increasing its expressiveness, and making it possible to write, e.g., *"Yesterday Alice knew the location of Bob on Saturday"*. The logic presented in Chapter VII subsumes that of Chapter VI.

On the other hand, in Chapter V we provide a runtime enforcement mechanism for dynamic privacy policies. The enforcement mechanism is implemented as a monitor that runs in parallel with the OSN. The monitor observes the events that occur in the system and switches on and off static privacy policies accordingly.

Unification of privacy policies and data

In Chapter VIII we study how to use Attribute-Based Encryption (ABE) [46, 79] to "stick" privacy policies to pictures in OSNs. Users in the OSN are assigned with attributes. An attribute in our setting is, for instance, being friends. Imagine that Alice is a friend of Bob, then Alice will have the attribute $\mathit{friend}(\mathit{Bob})$. When a user shares a picture, before uploading it to the OSN server, the system encrypts the picture in a way that only users with the specified attribute can see it. For example, Bob could specify that *"only my friends can see my pictures"*, that is, users that have the attribute $\mathit{friend}(\mathit{Bob})$. In the OSN database, the picture that is encrypted, therefore the OSN provider cannot visualise it. The encrypted picture can now move around the distributed database together with the policy. When another user access the resource, say Alice, she gives to the decryption algorithm all her attributes. If $\mathit{friend}(\mathit{Bob})$ is among them she will be able to visualise the picture, otherwise she will only access the ciphertext. Note that using this approach we solve *Problem iii)* (cf. Section I.1).

I.3.1 Thesis Outline

This section briefly outlines the content of each individual chapter and states the contribution of the author in the corresponding paper.

Chapter II: A Privacy Policy Framework for Social Networks. In this chapter we present a privacy policy framework for OSNs. The framework allows users to write static privacy policies that determine who can access their information. It consists of a formal model of the OSN, a (propositional) knowledge-based logic, and a formal privacy policy language based on the previous logic. The framework may be tailored by providing suitable instantiations of the different relationships, the information that users can know, and the additional facts or rules a particular OSN should satisfy. Besides, we provide instantiations of Facebook and Twitter in our formalism. We use the instantiations to model all the available privacy settings in Facebook and Twitter today. Furthermore, we provide a number of richer privacy policies which neither Facebook nor Twitter implement.

Statement of contributions: This paper was co-authored with Gerardo Schneider. Raúl was responsible for the development of the theory of social networks models, the knowledge-based language and the privacy policy language. Moreover, Raúl wrote the instantiation of Facebook and Twitter and all their privacy policies in the framework.

This paper was published in the proceedings of the *12th edition of the International Conference on Software Engineering and Formal Methods* (SEFM) 2014.

Chapter III: Formalising Privacy Policies in Social Networks. In this chapter we extend the framework presented in Chapter II. First, we lift the knowledge-based logic and privacy policy language from propositional to first-order, thus, increasing its expressiveness. Second, agents are enhanced with a reasoning engine allowing for the inference of knowledge from previously acquired knowledge. This allows us to encode possible inferences that users can perform when accessing various pieces of information. The engine uses the standard axioms from epistemic logic. To describe the evolution of the OSN, we use operational semantics rules. These rules are classified into four categories: epistemic, topological, policy, and hybrid, depending on whether the events under consideration change the knowledge of the OSN users, the structure of the social graph, the privacy policies, or a combination of the above, respectively. We formally introduce a notion of privacy preserving OSN. That is, an OSN where there is no sequence of events that can eventually violate a privacy policy. We provide specific rules for describing the behaviour of Twitter, and prove that it is privacy-preserving with respect to the set of privacy policies that the OSN offers today. We also show that Twitter and Facebook are not privacy-preserving in the presence of additional natural privacy policies.

Statement of contributions: This paper was co-authored with Musar Balliu and Gerardo Schneider. Raúl was responsible for the development of the theory regarding the extensions to the logic and the policy language; the dynamic part of the framework, that is, operational semantics rules, the underlying labelled transition system; and the concept of privacy-preserving OSN. Moreover, Raúl instantiated Twitter and Facebook and wrote the proofs of privacy-preservation in both OSNs.

This paper was published in the *Journal of Logical and Algebraic Methods in Programming* (JLAMP) 2017.

Chapter IV: Model Checking Social Network Models. Our privacy policy framework is based on a knowledge-based logic very similar to epistemic logic [29]. The semantics of epistemic logic makes use of Kripke models whereas our knowledge-based logic uses *social network models* (SNMs). SNMs are social graphs enriched with *knowledge bases* containing the information that the agents know. The properties of knowledge in epistemic knowledge have been studied for decades and are well understood. For SNMs, however, there is no characterisation of what knowledge properties hold. We show that the properties of knowledge that are sound with respect to Kripke models are also sound with respect to SNMs. We give a satisfaction-preserving encoding of SNMs into *canonical* Kripke models, and we also characterise which Kripke models may be translated into SNMs. We prove that the model checking problem for SNMs is decidable. Finally, we show that, for SNMs, the model checking problem is computationally cheaper than the one based on standard Kripke models.

Statement of contributions: This paper was co-authored with Gerardo Schneider. Raúl developed the transformation functions from Kripke models to SNM and vice versa; the proofs of soundness of the knowledge properties of epistemic logic and decidability of the model checking algorithm; and the formal complexity comparison of the model checking problem in SNMs and Kripke models.

This paper was published in the proceedings of the *8th International Symposium on Games, Automata, Logics, and Formal Verification* (GandALF) 2017.

Chapter V: An Automata-based Approach to Evolving Privacy Policies for Social Networks. In this chapter we present a runtime monitoring system that actives static privacy policies depending on *time* and *events*. In particular, we introduce a novel formalism, *policy automata*, which is a transition system where static privacy policies may be defined per state. The approach is policy agnostic, hence it can be applied to any OSN which has an enforcement mechanism for static privacy policies such as Facebook, Twitter, Instagram, Snapchat and so on. We provide a proof-of-concept implementation for the distributed social network Diaspora using the runtime verification tool LARVA to synthesise policy enforcement monitors.

Statement of contributions: This paper was co-authored with Christian Colombo, Gordon J. Pace and Gerardo Schneider. Raúl contributed to the definition of the policy automata theory, developed the proof-of-concept implementation in Diaspora, and wrote all the proofs in the paper.

This paper was published in the proceedings of the *16th International Conference on Runtime Verification* (RV) 2016.

Chapter VI: Specification of Evolving Privacy Policies for Online Social Networks. In this paper we extend the framework presented in Chapter III to express *dynamic* (and *recurrent*) privacy policies that are activated or deactivated by context (events) or time. First, we add temporal operators to the knowledge-based logic to express properties based on traces representing the evolution of the OSN. Second, we introduce a new *learning* modality that allows for reasoning about the moment in time when a user learnt something. Third, we extend the privacy policy language to specify intervals of time when the policies must be enforced, and also include the learning modality as part of the syntax. Policies, and formulae in the logic, are interpreted over (timed) traces representing the evolution of the OSN. Finally, we prove that checking privacy policy conformance, and the model-checking problem for the knowledge-based logic are decidable.

Statement of contributions: This paper was co-authored with Ivana Kellyérová, César Sánchez and Gerardo Schneider. Raúl was responsible for the development of the extensions to the logic, the privacy policy language, and the notion of traces. He also contributed to the decidability proofs.

This paper was published in the proceedings of the *23rd International Symposium on Temporal Representation and Reasoning* (TIME) 2016.

Chapter VII: Timed Epistemic Knowledge Bases for Social Networks. In this chapter we present a version of the framework presented in Chapter III which includes a knowledge-based logic that allows for reasoning not only about information available to the different agents, but also about the moments at which events happen and new knowledge is acquired or deduced. To do this, we add time-stamps in predicates and modalities. Time-stamped predicates indicate the *active* time of a piece of information, e.g., *"location of Bob on Monday at 20:00"*. Time-stamped modalities indicate when users know something, for instance, *"Yesterday Alice know the location of Bob on Monday at 20:00"*. In this logic we include both an epistemic modality and a belief modality. By using these operators we can differentiate between true knowledge and beliefs that come from disclosure of information that may not be accurate.

As in Chapters II,III and VI, our ultimate goal is to develop a privacy policy language. The logic described in this chapter is used as a basis for a privacy policy language more expressive than the ones presented in previous chapters. Furthermore, the logic presented in this chapter subsumes that of VI. We show that the learning modality and temporal operators introduced in Chapter VI become derived operators in this logic. Policies and formulae in the logic are interpreted over timed traces representing the evolution of the OSN.

Finally, we present an algorithm for deducing knowledge, which can also be instantiated with different variants of how the epistemic information is preserved over time. This algorithm is used to model the deductive engine of the agents in our framework.

Statement of contributions: This paper was co-authored with César Sánchez and Gerardo Schneider. Raúl was responsible for the definition of the new knowledge-based logic, privacy policy language and the timed deduction rules for users knowledge.

This paper is currently under submission.

Chapter VIII: Secure Photo Sharing in Social Networks. In this chapter we propose a practical solution to secure photo sharing in OSNs. Our approach assumes nothing about the architecture of the OSN which can be either centralised or distributed. This solution solves the inconsistencies that appear in distributed OSNs—and distributed data stores for centralised OSNs—as a consequence of treating photos and access policies separately. For instance, when updating an access policy, it is not instantly replicated in all the nodes of the system; some nodes may contain the old access policy, and, consequently, it is possible to disclose information to an undesired audience. We propose a solution to this problem based on attaching access policies to photos so that, each time a photo is re-shared, the access policy travels together with the photo. In order to attach a policy to a photo we use Attribute-based Encryption. Attributes in OSNs are the social relationships between users that are defined in the social graph.

We tested our solution on the distributed OSN Diaspora. We tried different configurations: one server (centralised mode) and more than three servers (decentralised mode). We also used different number of attributes to check whether the system's performance deteriorates as the number of attributes increases. We consider OSNs with a hundred attributes and up to twelve attributes per access policy. The evaluation shows that our solution can encrypt/decrypt photos in less than two seconds.

Statement of contributions: This paper was co-authored with Pablo Picazo-Sánchez and Gerardo Schneider. Raúl contributed to the design of the system and implemented the proof-of-concept prototype in Diaspora.

This paper was published in the proceedings of the *32nd International Conference on ICT Systems Security and Privacy Protection* (IFIP SEC) 2017.

Chapter II

A Formal Privacy Policy Framework for Social Networks

Raúl Pardo and Gerardo Schneider

Abstract. Social networks (SN) provide a great opportunity to help people interact with each other in different ways depending on the kind of relationship that links them. One of the aims of SN is to be flexible in the way one shares information, being as permissive as possible in how people communicate and disseminate information. While preserving the spirit of SN, users would like to be sure that their privacy is not compromised. One way to do so is by providing users with means to define their own privacy policies and give guarantees that they will be respected. In this paper we present a privacy policy framework for SN, consisting of a formal model of SN, a knowledge-based logic, and a formal privacy policy language. The framework may be tailored by providing suitable instantiations of the different relationships, the events, the propositions representing what is to be known, and the additional facts or rules a particular social network should satisfy. Besides, models of Facebook and Twitter are instantiated in our formalism, and we provide instantiations of a number of richer privacy policies.

II.1 Introduction

A *social network* is a structure made up of a set of *agents* (individuals or organisations), which are connected via different kinds of relationships. People and organisations use social networks (SN) to interact on a peer-to-peer manner and also to broadcast information related to themselves or others with selected subgroups of other agents. Users expect that social network services (SNS) provide flexibility and easy-to-use interfaces for achieving the intended objectives in a fast and reliable manner. This flexibility, however, comes with the potential problem of compromising organisations' and individuals' privacy.

Privacy in SN may be compromised in different ways: from direct observation of what is posted (seen by non-allowed agents), by inferring properties of data (*metadata privacy leakages*), indirectly from the topology of the SN (e.g., knowing who our friends are), to more elaborate intentional attackers such as *sniffers* or *harvesters* [35]. In this paper we are mainly concerned with the first 3 kinds of privacy issues. In order to tackle them, we look into the problem of defining a formal language for writing rich privacy policies in the context of social networks. We aim at defining a privacy policy language able to express at least the following (kinds of) policies: i) All privacy policies currently supported by existing SN like Facebook; ii) Privacy policies describing properties on attributes, i.e. not only coarse-grained properties as the fact that someone has post something, but about the content of the post itself; iii) Conditional privacy policies, which depend on the amount of current knowledge or permissions in the SN; iv) Privacy policies based on knowledge in a group of agents and distributed knowledge among several agents.

In order to achieve the above we propose a solution based on the definition of a rather general privacy policy framework that may be specialised for concrete SN instances. More concretely, our contributions are:

1. We propose a formal *privacy policy framework* consisting of: i) a generic model for social networks, formalised as a combination of hyper-graphs and Kripke structures; ii) the syntax and semantics of a knowledge-based logic to reason about the social network and privacy policies; iii) a formal language to describe privacy policies (based on the logic mentioned above), together with a conformance relation to be able to state whether a certain social network satisfies a given policy. (Section IV.2.2.)

2. We specify how the above privacy policy framework may be instantiated in order to be used in practice. (Section III.2.4.)

3. Our definition of *instantiated privacy policy framework* allows us to model not only

existing SN with their corresponding privacy policies, but also richer ones. We show the expressiveness of our approach by presenting instantiations of Twitter, Facebook, and richer privacy policies. (Section II.4.)

II.2 Privacy Policy Framework

In this section we define $\mathcal{PPF}$, a formal *privacy policy framework* for social networks. The framework is not only able to deal with explicit disclosure of information, but it also is equipped with internal machinery for detecting implicit knowledge.

Definition 1. *The tuple* $\langle \mathcal{SN}, \mathcal{KBL}, \vDash, \mathcal{PPL}, \vDash_C \rangle$ *is a* privacy policy framework *(denoted by* $\mathcal{PPF}$*), where*

- $\mathcal{SN}$ *is a social network model;*
- $\mathcal{KBL}$ *is a knowledge-based logic;*
- $\vDash$ *is a satisfaction relation defined for* $\mathcal{KBL}$*;*
- $\mathcal{PPL}$ *is a privacy policy language;*
- $\vDash_C$ *is a conformance relation defined for* $\mathcal{PPL}$*.* □

In what follows we define in more detail each of the components of $\mathcal{PPF}$.

II.2.1 The Social Network Model $\mathcal{SN}$

$\mathcal{SN}$ is a generic model for social networks representing the topology of the social network, modelling the different *connections* between agents, their knowledge, and the actions they are allowed to perform.

Preliminaries.. Before providing the definition of $\mathcal{SN}$ let us define Ag to be a finite and nonempty set of *agents*, $\mathcal{C}$ a finite and nonempty set of *connections*, representing the relations between agents (e.g. friendship, colleague, blocked, restricted), and Σ a finite and nonempty set of *actions*, representing what is allowed to be performed by the agents (e.g. posting, looking up an agent). Also, let Π be a finite set of privacy policies defined by

$$\Pi = \{\llbracket \psi_j \rrbracket_i \mid i \in Ag,\ j \in \{1, 2, \ldots, n_i\} \text{ and } \psi_j \in \mathcal{PPL}\}$$

containing all the privacy policies for each agent i (there are n_i privacy policies for each agent i, if $n_i = 0$ then there is no privacy policy associated with agent i).

Definition 2. *Given a nonempty set of propositions $\mathcal{P}$, we define a* social network model *$\mathcal{SN}$ to be a hypergraph of the form $\langle W, \{R_i\}_{i\in\mathcal{C}}, \{A_i\}_{i\in\Sigma}, \nu, KB, \pi\rangle$, where*

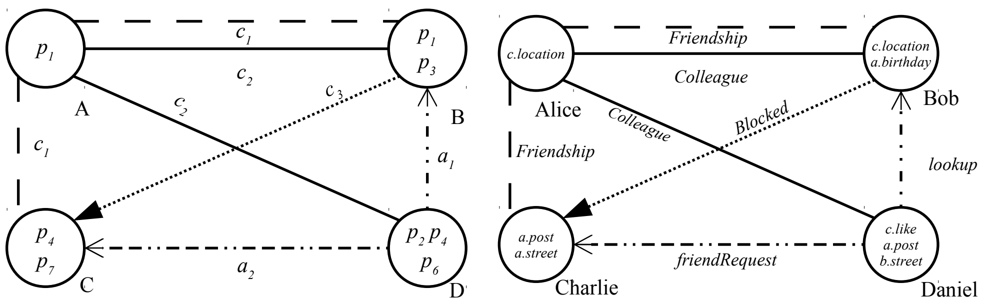
- *W is a nonempty set of* possible worlds*. Every world represents one of the agents defined in the set Ag.*
- *$\{R_i\}_{i\in\mathcal{C}}$ is a family of binary relations $R_i \subseteq W \times W$, indexed by connections. Given agents $x, y \in W$, we write xR_iy iff $(x, y) \in R_i$.*
- *$\{A_i\}_{i\in\Sigma}$ is a family of binary relations $A_i \subseteq W \times W$, indexed by actions. Given agents $x, y \in W$, we write xA_iy iff $(x, y) \in A_i$.*
- *ν is a valuation function returning the set of propositions which are true in a given world (i.e. $\nu : W \to 2^{\mathcal{P}}$).*
- *KB is a function giving the set of accumulated non-trivial knowledge for each agent, stored in what we call the* knowledge base *of the agent.* [1]
- *π is a function returning the set of privacy policies defined for a given agent (i.e. $\pi : W \to 2^{\Pi}$).* $\square$

We define a bijective function between agents and worlds $AW : Ag \to W$; hereafter we will interchangeably refer to elements of W as *worlds* or *agents*. We will sometimes use the indexes to denote the corresponding connections. So, given $\mathcal{C}$ to be the set $\{Friendship, Colleague\}$, then instead of writing $R_{Friendship}$ and $R_{Colleague}$ we will write $Friendship$ and $Colleague$ respectively. In addition we define $SN|_c$ to be the *projection over the connection* $c \in \mathcal{C}$ for a given social network model SN, as the graph $SN|_c = \langle W, R_c\rangle$, where W is the set of worlds of SN and R_c is the binary relation defined in SN for the connection c. Finally, given a set of agents $G \subseteq Ag$ and a projection $SN|_c$, we define the following predicate $clique(SN|_c, G)$ iff $\forall i, j \in G.\ iR_cj \wedge jR_ci$.

Example 1. *We illustrate how a small fragment of a generic social network could be modelled according to definition 2. The $\mathcal{SN}$ consists of: i) 4 agents, $Ag = \{A, B, C, D\}$; ii) a set of 3 connections, $\mathcal{C} = \{c_1, c_2, c_3\}$; iii) the set $\Sigma = \{a_1, a_2\}$, representing the actions allowed among users.*

A graphical representation of the defined social network is given in Fig. II.1a. The dashed line and the plain line represent the c_1 and c_2 relations, respectively. They are not directed because we assume these relations are symmetric. On the other hand, the c_3 relation (represented by a dotted line) relates only B and C, and it is directed.

[1] We will formally define this function in subsection III.2.2, since its definition requires a formal specification of $\mathcal{KBL}$ subformulae.



(a) Example of a generic $\mathcal{SN}$ (b) $\mathcal{FPPF}_{\mathcal{I}}$ of a Facebook-like SN

Figure II.1: Examples of social network models

The allowed actions are represented by the dashed and dotted directed arrows. Actions represent interaction between 2 agents. In the example, action a_1 has D as source and B as target. Associated to each world there is a set of propositions over $\{p_1, p_2, \ldots, p_7\} \subseteq \mathcal{P}$ explicitly representing basic knowledge of the agent. For instance, in Fig. II.1a it is shown that agent C knows p_4 and p_7. □

II.2.2 The knowledge-based logic for social networks $\mathcal{KBL}$

We define here a logic for representing and reasoning about knowledge. We give semantics to the logic $\mathcal{KBL}$ over a knowledge-based representation built on top of the social network model $\mathcal{SN}$.

Definition 3. *Given $i, j \in Ag$, $a \in \Sigma$, $p \in \mathcal{P}$, and $G \subseteq Ag$, the* knowledge-based logic *$\mathcal{KBL}$ is inductively defined as:*

$$\begin{array}{rcl}
\gamma & ::= & \neg\gamma \mid \gamma \wedge \gamma \mid \psi \mid \varphi \\
\psi & ::= & P_i^j a \mid GP_G^j a \mid SP_G^j a \\
\varphi & ::= & p \mid \varphi \wedge \varphi \mid \neg\varphi \mid K_i\varphi \mid E_G\varphi \mid S_G\varphi \mid D_G\varphi.
\end{array}$$

The intuitive meaning of the modalities is as follows.

- $K_i\varphi$ (Basic knowledge): Agent i knows φ.
- $E_G\varphi$ (Everyone knows): Every agent in the group G knows φ.
- $S_G\varphi$ (Someone knows): At least one agent in the group G knows φ.
- $D_G\varphi$ (Distributed knowledge): φ is distributed knowledge in the group of agents G (i.e. the combination of individual knowledge of the agents in G).

$SN, u \models \neg p$	iff	$\neg p \in \nu(u)$
$SN, u \models p$	iff	$p \in \nu(u)$
$SN, u \models \neg\varphi$	iff	$SN, u \not\models \varphi$
$SN, u \models \varphi \wedge \psi$	iff	$SN, u \models \varphi$ and $SN, u \models \psi$
$SN, u \models K_i\delta$	iff	$\begin{cases} \delta \in KB(i) & \text{if } \delta = K_j\delta', \text{where } j \in Ag \\ SN, i \models \delta & \text{otherwise} \end{cases}$
$SN, u \models P_i^j a$	iff	$(i, j) \in A_a$
$SN, u \models GP_G^j a$	iff	$(n, j) \in A_a$ for all $n \in G$
$SN, u \models SP_G^j a$	iff	there exits $n \in G$ such that $(n, j) \in A_a$
$SN, u \models S_G\delta$	iff	there exits $i \in G$ such that $SN, i \models K_i\delta$
$SN, u \models E_G\delta$	iff	$SN, i \models K_i\delta$ for all $i \in G$
$SN, u \models D_G\delta$	iff	$\begin{cases} SN, u \models S_G\delta' \text{ and } SN, u \models S_G\delta'' & \text{if } \delta = \delta' \wedge \delta'' \\ SN, u \models S_G\delta & \text{otherwise} \end{cases}$

Table II.1: $\mathcal{KBL}$ satisfaction relation

- $P_i^j a$ (Permission): Agent i is allowed to perform action a to agent j.
- $GP_G^j a$ (Global Permission): All agents specified in G are allowed to perform action a to agent j.
- $SP_G^j a$ (Someone is Permitted): At least one agent specified in G is allowed to perform action a to agent j.

We will denote with $\mathcal{F}_{\mathcal{KBL}}$ the set of all well-formed formulae of $\mathcal{KBL}$ as defined by the grammar given in above definition. Similarly, $\mathcal{F}_{\mathcal{KBL}}^{\mathcal{K}}$ will denote those defined by the syntactic category φ and $\mathcal{F}_{\mathcal{KBL}}^{\mathcal{P}}$ will denote the subformulae of the logic defined by the syntactic category ψ. The function giving the knowledge base of an agent, informally described in section II.2.1, has the following type $KB : Ag \rightarrow 2^{\mathcal{F}_{\mathcal{KBL}}^{\mathcal{K}}}$. We define in what follows the satisfaction relation for $\mathcal{KBL}$ formulae.

Definition 4. *Given a $SN = \langle W, \{R_i\}_{i\in\mathcal{C}}, \{A_i\}_{i\in\Sigma}, \nu, KB, \pi\rangle$, the agents $i, j, u \in Ag$, a finite set of agents $G \subseteq Ag$, an action $a \in \Sigma$, $\delta \in \mathcal{F}_{\mathcal{KBL}}^{\mathcal{K}}$, and $\varphi, \psi \in \mathcal{F}_{\mathcal{KBL}}$, the* satisfaction relation $\models$ *is defined as shown in Table II.1.* □

Note that we explicitly add the negation of a proposition. It represents knowing the negation of a fact (e.g $K_i\neg p$) which is different than not knowing it (i.e. $\neg K_i p$). Moreover, it is important to point out that $\mathcal{KBL}$ is not minimal as the last 5 modalities can be defined in terms of more basic cases as follows: $S_G\delta \triangleq \bigvee_{i\in G} K_i\delta$, $E_G\delta \triangleq \bigwedge_{i\in G} K_i\delta$, $GP_G^j a \triangleq \bigwedge_{i\in G} P_i^j a$, $SP_G^j a \triangleq \bigvee_{i\in G} P_i^j a$ and $D_G\delta$ is already defined in terms of S_G as shown in its semantical definition. Note that as the set G is finite, so are the disjunction and the conjunction for S_G, E_G, GP_G^j and SP_G^j.

Example 2. *$\mathcal{KBL}$ enables the possibility of reasoning about epistemic and deontic properties. As stated in $\mathcal{SN}$ showed in Example 1, D is allowed to execute a_1, which will affect B. In $\mathcal{KBL}$ we can formally check the previous statement by checking satisfaction of the following judgement: $SN, B \vDash P_D^B\ a_1$.*

We can also build more complex expressions in which we actually leverage the reasoning power of $\mathcal{KBL}$. For instance, we can check whether the following holds for agent A:

$$SN, A \vDash \neg K_B\ p_1 \wedge \neg K_C K_A\ p_1 \implies \neg SP_{\{B,C\}}^A\ a_1,$$

which means that if agent B does not know p_1 and agent C does not know that agent A knows p_1 then it is not permitted for any of the agents B and C to execute the action a_1 to the agent A. $\square$

Apart from checking properties in the model, $\mathcal{KBL}$ also permits to reason about certain properties that hold in general. Given a social network SN, $i, j \in Ag$, and formulae $\varphi, \psi \in \mathcal{F}_{\mathcal{KBL}}^{\mathcal{K}}$, we can state and prove the following lemma on the influence of the individuals knowledge and their combination as distributed knowledge.

Lemma 1. *$SN, i \vDash K_i\varphi \wedge K_j\psi \implies D_{\{i,j\}}\varphi \wedge \psi$.* $\square$

II.2.3 The privacy policy language for social networks $\mathcal{PPL}$

$\mathcal{KBL}$ is an expressive language for specifying and reasoning about epistemic and deontic properties of agents in SN models. However, the language is not completely suitable for writing privacy policies, and thus a different language is needed for this purpose. Privacy policies in social networks can be seen as explicit statements in which agents specify what cannot be known about them or what is not permitted to be executed. The syntax of the privacy policy language $\mathcal{PPL}$ is based on that of $\mathcal{KBL}$, but adapted to express privacy policies.

Definition 5. *Given the agents $i, j \in Ag$ and a nonempty set of agents $G \subseteq Ag$, the syntax of the* privacy policy language *$\mathcal{PPL}$ is inductively defined as follows:*

$$\begin{array}{rcl}
\delta & ::= & \delta \wedge \delta \mid [\![\varphi \implies \neg\psi]\!]_i \mid [\![\neg\psi]\!]_i \\
\varphi & ::= & \psi \mid \neg\psi \mid \varphi \wedge \varphi \\
\psi & ::= & E_G\gamma \mid S_G\gamma \mid D_G\gamma \mid K_i\gamma \mid GP_G^j a \mid SP_G^j a \mid P_i^j a \mid \psi \wedge \psi. \\
\gamma & ::= & p \mid \gamma \wedge \gamma
\end{array}$$

$SN \vDash_C \tau_1 \wedge \tau_2$	iff	$SN \vDash_C \tau_1 \wedge SN \vDash_C \tau_2$
$SN \vDash_C \llbracket \neg\psi \rrbracket_i$	iff	$SN, i \vDash \neg\psi$
$SN \vDash_C \llbracket \varphi \implies \neg\psi \rrbracket_i$	iff	$SN, i \vDash \varphi$ then $SN \vDash_C \llbracket \neg\psi \rrbracket_i$

Table II.2: $\mathcal{PPL}$ conformance relation

$\mathcal{PPL}$ may be seen as formed by a subset of formulae definable in $\mathcal{KBL}$ wrapped with the $\llbracket\ \rrbracket_i$ operator, specifying which agent has defined the privacy policy. As before, we define $\mathcal{F}_{\mathcal{PPL}}$ to be the set of $\mathcal{PPL}$ well-formed formulae defined as given by the grammar in the above definition. A basic privacy policy for an agent i, given by δ in definition 9, is either a direct restriction ($\llbracket \neg\psi \rrbracket_i$) or a conditional restriction ($\llbracket \varphi \implies \neg\psi \rrbracket_i$). $\mathcal{F}^{\mathcal{C}}_{\mathcal{PPL}}$ will denote sbuformulae belonging to the syntactic category φ (conditions) and $\mathcal{F}^{\mathcal{R}}_{\mathcal{PPL}}$ subformulae of the syntactic category ψ (restrictions). Instead of defining a satisfaction relation for $\mathcal{PPL}$, we define the following *conformance* relation to determine when a $\mathcal{SN}$ respects a given privacy policy.

Definition 6. *Given a* $SN = \langle W, \{R_i\}_{i\in\mathcal{C}}, \{A_i\}_{i\in\Sigma}, \nu, KB, \pi\rangle$*, an agent* $i \in Ag$*,* $\varphi \in \mathcal{F}^{\mathcal{C}}_{\mathcal{PPL}}$*,* $\psi \in \mathcal{F}^{\mathcal{R}}_{\mathcal{PPL}}$ *and* $\tau_1, \tau_2 \in \mathcal{F}_{\mathcal{PPL}}$*; the* conformance relation $\vDash_C$ *is defined as shown in Table III.2.* □

Example 3. *The following are the privacy policies for agent A (cf. Example 1):* $\pi(A) = \{\llbracket \neg S_{\{B,C,D\}}\ p_1 \rrbracket_A, \llbracket K_B\ p_1 \implies \neg P^A_B\ a_1 \rrbracket_A\}$*. The intuitive meaning of the first policy is that nobody can know* p_1 *(apart from A who is the only agent left in the* $\mathcal{SN}$*). The second one means that if agent B knows* p_1 *then she is not permitted to execute the action* a_1 *to A.* □

II.3 $\mathcal{PPF}$ instantiation

In the previous section we have presented a generic framework for defining privacy policies in social networks. In order to be usable, the framework needs to be instantiated, as specified in the following definition.

Definition 7. *We say that a* $\mathcal{PPF}$ *is an* instantiated privacy policy framework *iff an instantiation for the following is provided:*

- *The set of agents* Ag*;*
- *The set of propositions* $\mathcal{P}$ *(*$p \in \mathcal{P}$ *may be given a structure);*
- *The set of connections* $\mathcal{C}$*;*

- *The set of auxiliary functions over the above connections;*
- *The set of actions Σ;*
- *A set of properties written in $\mathcal{KBL}$ (these properties may be seen as assumptions on the social network);*
- *A set of constraints over the policies defined in the language $\mathcal{PPL}$.* $\square$

We write $\mathcal{FPPF}_{Name}$ for the instantiation of a $\mathcal{PPF}$ on a specific social network *Name*. In what follows we show an example of instantiation.

Example 4. *We present here $\mathcal{FPPF}_{FBook\text{-}like}$, an instantiation of the privacy policy framework given in Example 1 for a Facebook-like social network. (Fig. II.1b shows the $\mathcal{SN}$ for the instantiated $\mathcal{FPPF}$.)*

Agents *We redefine the set of agents to be $Ag = \{Alice, Bob, Charlie, Daniel\}$.*

Propositions *We define a structure for the propositions, by requiring them to be of the form owner.attribute (e.g. Alice.street).*

Connections. *In this particular instantiation we consider only the following connections: $\mathcal{C} = \{Friendship, Colleague, Blocked\}$.*

Auxiliary functions. *The following auxiliary functions (from Ag to 2^{Ag}) will help to retrieve the corresponding sets associated to the above defined connections: $friends(i) = \{u \mid iR_{Friendship}u \text{ and } uR_{Friendship}i\}$; $blocked(i) = \{u \mid iR_{Blocked}u\}$; $colleagues(i) = \{u \mid iR_{Colleague}u \text{ and } uR_{Colleague}i\}$. These functions are notably useful when writing formulae (both in $\mathcal{KBL}$ and $\mathcal{PPL}$), since it allows to refer to groups of agents defined by their relationships.*

Actions. *The set of actions is instantiated as $\Sigma = \{sendRequest, lookup\}$.*

Assumptions on the $\mathcal{SN}$. *Different social networks are characterised by different properties. We use $\mathcal{KBL}$ for defining these properties (or assumptions). In a Facebook-like social network some attributes are a composition of others. We introduce here the notion of* record, *that is a complex attribute composed by others. We assume that the attribute location of an agent is composed by the following attributes: street, country, and city. Given agents $u, i, j, h \in Ag$ and the group $G = \{i, j, h\}$ we assume the following property holds:*

$$SN, i \models S_G\ u.country \wedge S_G\ u.city \wedge S_G\ u.street \implies D_G\ u.location \qquad \text{(II.1)}$$

moreover if $i = j = h$ we can derive the following property:

$$SN, i \models K_i(u.country \wedge u.city \wedge u.street) \implies K_i\ u.location \qquad \text{(II.2)}$$

In addition we can also model facts that we assume to be true in the social network. For example, we could assume that if some information is distributed knowledge among users who are friends, then this information becomes known to all of them individually. Formally we say that given a set of agents $G \subseteq Ag$, an agent $u \in Ag$ and a formula $\varphi \in \mathcal{F}^{\mathcal{K}}_{\mathcal{KBL}}$, for all $i \in G$ the following holds:

$$\text{if } SN, u \vDash D_G\varphi \text{ and } clique(SN|_{Friendship}, G) \text{ then } SN, i \vDash K_i\varphi. \qquad \text{(II.3)}$$

Constraints over privacy policies. *A common constraint in social networks is that agents can only write policies about their own data. In $\mathcal{PPL}$ it is possible to write $[\![\neg K_j\ u.attribute]\!]_i$ where $i, j, u \in Ag$ and $i \neq j \neq u$. This policy, defined by agent i, forbids agent j to know attribute from agent u. Agent i is thus constraining the accessibility of certain information about an agent other than herself. To solve this we could add the following constraint: Given an agent $i \in Ag$ and her privacy policies, $\bigwedge_{j \in 1,\ldots,n} \tau_j \in \pi(i)$, where $\tau_j = [\![\varphi]\!]_i$, if $\varphi = \neg\varphi'$ or $\varphi = \varphi'' \implies \neg\varphi'$ then it is not permitted that $u.attribute \in \varphi'$ for any $u \in Ag$. $u \neq i$, meaning that agents can only define policies about their own data. Likewise, users should not be able to write permission restrictions over other users. In order to address this issue we extend the previous restriction with the following: given an agent $j \in Ag$, an action $a \in \Sigma$ and the set $G \subseteq Ag$, it is not the case for $i \neq j$ that $P^j_u a$, $SP^j_G a$ or $GP^j_G a \in \varphi'$.* □

For a given instantiation we could prove more properties besides the ones given as assumptions. The following lemma exemplifies the kind of properties we can prove about instantiated privacy policy frameworks in general and for $\mathcal{FPPF}_{\text{FBook-like}}$ in particular.

Lemma 2. *Given $u \in Ag$, if $SN, u \vDash D_G$ $(u.country \wedge u.city \wedge u.street)$, and assuming the group of agents $G \subseteq Ag$ are all friends to each other (i.e. $clique(SN|_{Friendship}, G)$), then $SN \nvDash_C [\![\neg S_{\{Ag \setminus \{u\}\}} u.location]\!]_u$.* □

II.4 Case Studies

$\mathcal{PPF}$ may be instantiated for various social networks. We show here how to instantiate Twitter and Facebook. Though our formalisation is expressive enough to fully instantiate the social networks under consideration, due to lack of space we will only show minimal instantiations which allow us to represent all the existing privacy policies in the mentioned social networks.

Before going into the details of our instantiation, we describe some preliminaries. In the rest of the section we will use i, j, u to denote agents ($i, j, u \in Ag$), and G to denote a finite subset of agents ($G \subseteq Ag$), where Ag is the set of agents registered in the instantiated social network. Given an attribute att of an agent u (denoted by $u.att$), we will sometimes need to distinguish between different occurrences of such an attribute. In that case we will write $u.att_\eta$ ($\eta \in \{1, \ldots, n_u\}$, with n_u being the maximum number of occurrences of the attribute; by convention, if there are no occurrence of $u.att$, we have that $n_u = 0$). For example, if we assume that agent u's location changes and we want to refer to these different locations we will write $u.location_\eta$.

II.4.1 Twitter privacy policies

Twitter is a microblogging social network. Users share information according to the connections established by the *follower* relationship, which permits (depending on the privacy policies) a user to access the tweets posted (or tweeted) from the followed user. Users interact by posting (or tweeting) 140 characters long messages called *tweets*. Let us define the instantiation $\mathcal{FPPF}_{\text{Twitter}}$ as follows.

Propositions. The proposition in $\mathcal{FPPF}_{\text{Twitter}}$ are defined by the set

$\mathcal{P} = \{owner.email, owner.location_i, owner.tweet_j, owner.retweet_{tweetRef}\}$

where $owner \in Ag$, and attributes are the following: *email*, is the user's email; $location_\eta$ represents a location of a given user ; $tweet_\eta$ the tweets a given user has tweeted; and $retweet_{tweetRef}$ representing the fact or retweeting (or sharing a tweet already tweeted by another user) where *tweetRef* is the reference to the original tweet.

Connections. The set of connections only includes the follower relationship, $\mathcal{C} = \{Follower\}$.

Auxiliary functions. We define the function

- $followers(i) = \{u \mid u \in Ag \wedge iR_{Follower}u\}$

which returns all the agents u who i is following.

Actions. Actions are defined as $\Sigma = \{tweet, lookup, sendAd\}$, where *tweet* represents tweeting (posting a tweet), *lookup* represents the possibility of reaching a user's profile and *sendAd* sending an advertisement to a user.

Twitter does not have a large amount of privacy policies since the aim is to make information accessible to as many people as possible. Yet there are important considerations concerning privacy. These policies are specified in $\mathcal{FPPF}_{\text{Twitter}}$ as follows.

- Protect my Tweets: Two cases: i) Only those in u's group of followers can see her tweets: $[\![\neg S_{\{Ag \setminus followers(u) \setminus \{u\}\}}\ u.tweet_\eta]\!]_u$; ii) Only u's followers may see her retweets: $[\![\neg S_{\{Ag \setminus followers(u) \setminus \{u\}\}} u.retweet_{tweetRef}]\!]_u$.
- Add my location to my tweets: Twitter provides the option of adding the agents' location to their tweets. The following policy specifies that nobody can see the

user's locations: $\llbracket \neg S_{\{Ag\setminus\{u\}\}}\ u.location_\eta \rrbracket_u$.

- Let others find me by my email address: $\llbracket \neg K_i\ u.email \implies \neg P_i^u\ lookup \rrbracket_u$, meaning that if an agent i does not know u's email, then she is not allowed to find u by looking her up.
- Tailor ads based on information shared by ad partners: Assuming G to be the group of ads partners, the policy is defined as $\llbracket \neg SP_G^u\ sendAd \rrbracket_u$, meaning that none of the advertisement companies taking part in the system is able to send advertisements to user u.

II.4.2 Facebook privacy policies

Facebook is a social network system in which people share information by means of posts. Each user owns a *timeline* which contains all her posts and information about the main events which can be handled by the social network (e.g. birthday, new relationships, attendance to events). The main connection between users is *friendship*, though it is possible to create special relations.

We show here the instantiation $\mathcal{FPPF}_{\text{Facebook}}$. Since we are modelling just the parts of Facebook relevant for defining privacy policies, we borrow the set $\mathcal{C}$ from the instantiation presented in Example 4. We also borrow the set of auxiliary functions and we add $friends^2(i) = \bigcup_{j \in friends(i)} friends(j)$; it allows us to write formulae about friends of friends. We extend the set Σ with the action *postTimeline* (representing posting on a user's timeline), and the actions *inviteEvent*, *inviteGroup* and *tag*, which represent sending an invitation to join an event or a group and being tagged on a picture. As for the structure of the propositions, we define the set $\mathcal{P} = \{owner.post^j_\eta, owner.like_\eta, owner.location, owner.phone, owner.email\}$ where $owner \in Ag$, $owner.post^j_\eta$ represents the posts *owner* posted on the j's timeline (e.g. $Alice.post^{Bob}_1$ is the first post of Alice in Bob's timeline), $owner.like_\eta$ are the posts *owner* has liked and *owner.location*, *owner.phone* and *owner.email* are, respectively, the actual location, phone and the email attributes of *owner*. Similarly to $\mathcal{FPPF}_{\text{Twitter}}$, we do not specify properties for the $\mathcal{SN}$ nor restrictions over policies.

Privacy Settings and Tools

In what follows we go through all privacy policies a user can define in the section 'Privacy and Tools' of Facebook, and we provide their formalisation in $\mathcal{FPPF}_{\text{Facebook}}$. The policies are defined depending on the set of users which they affect.

Who can see my stuff?. In the first section Facebook enables users to set a default audience for their posts. In $\mathcal{FPPF}_{\text{Facebook}}$ we can formally specify these restrictions as follows:

- (Public) no policy since everyone is able to access the posts;
- (Friends) $[\![\neg S_{\{Ag \setminus friends(u) \setminus \{u\}\}} u.post_n^j]\!]_u$;
- (Only me) $[\![\neg S_{\{Ag \setminus \{u\}\}} u.post_n^j]\!]_u$;
- (Custom) $[\![\neg S_{\{G\}} u.post_n^j]\!]_u$.

The intuition behind these policies is specifying the group of agents who are not allowed to know the information about u's posts.

Who can contact me?. In a second section users are provided with the possibility of deciding who can send them friend requests:

- (Everyone) No need of privacy policy;
- (Friends of Friends) $[\![\neg SP^u_{\{Ag \setminus friends^2(u)\}} request]\!]_u$;

note that we specify who cannot send the friend request, which in this case are the agents who are not in the group of friends of friends.

Who can look me up?. Finally, a user can be looked up by its email address or phone number. Given $a \in \{phone, email\}$, specified as

- (Everyone) No privacy policy is needed;
- (Friends of Friends) $[\![(\neg K_i\ u.a \implies \neg P_i^u\ lookup)]\!]_u$ where $i \in friends^2(u)$ and $[\![(\neg SP^u_{\{Ag \setminus friends^2(u) \setminus \{u\}\}} lookup)]\!]_u$;
- (Friends) $[\![\neg K_i\ u.a \implies \neg P_i^u\ lookup]\!]_u \wedge [\![\neg SP^u_{\{Ag \setminus friends(u) \setminus \{u\}\}} lookup]\!]_u$, where $i \in friends(u)$.

Timeline and Tagging

Besides the previous policies, Facebook allows to define a set of policies related with who can post on our wall and how to manage our tags. We show now their formalisation in $\mathcal{FPPF}_{\text{Facebook}}$.

Who can post on my timeline.. Facebook offers the possibility of controlling the people allowed to write in a user's wall:

- (Only me) $[\![\neg SP^u_{\{Ag \setminus \{u\}\}} postTimeline]\!]_u$;
- (Friends) $[\![\neg SP^u_{\{Ag \setminus friends(u) \setminus \{u\}\}} postTimeline]\!]_u$.

Who can see things on my timeline?. In Facebook it is possible to establish a bounded audience for the posts located in a user's wall. We formally define the privacy policies as:

- (Everyone) No privacy policy needed;
- (Friends of friends) $[\![\neg S_{\{Ag \setminus friends(u) \setminus friends^2(u) \setminus \{u\}\}} i.post^u_n]\!]_u$ (implicitly includes friends);
- (Friends) $[\![\neg S_{\{Ag \setminus friends(u) \setminus \{u\}\}} i.post^u_n]\!]_u$;
- (Only me) $[\![\neg S_{\{Ag \setminus \{u\}\}} i.post^u_n]\!]_u$;
- (Custom) $[\![\neg S_{\{Ag \setminus G \setminus \{u\}\}} i.post^u_n]\!]_u$.

Manage blocking

Facebook offers the possibility of restricting or blocking the access to our information to a predefined set of users. It also allows to block users from doing more concrete actions as sending apps invitations, events invitations or apps. This is done by defining blocked and restricted users. Since these policies are similar, we define only the policies related with blocked users. Facebook defines blocking as *'Once you block someone, that person can no longer see things you post on your timeline, tag you, invite you to events or groups, start a conversation with you, or add you as a friend'* we formally define the previous statement in $\mathcal{FPPF}_{\text{Facebook}}$ with the following set of privacy policies. A blocked user cannot: i) see things you post on your time line: $[\![\neg S_{Blocked(u)}\ u.post^u_\eta]\!]_u$; ii) tag you: $[\![\neg SP^u_{Blocked(u)}\ tag]\!]_u$; iii) invite to events or to join groups: $[\![\neg SP^u_{Blocked(u)}\ inviteEvent]\!]_u \wedge [\![\neg SP^u_{Blocked(u)}\ inviteGroup]\!]_u$; iv) send a friend request: $[\![\neg SP^u_{Blocked(u)}\ sendRequest]\!]_u$.

II.4.3 More complex policies

We have shown how to specify all the privacy policies of Twitter and Facebook. We show here how to express other policies, which the aforementioned SN do not offer.

A more expressive language

One of the advantages of $\mathcal{PPF}$ is its flexibility when defining the structure of the propositions. It allows us to talk about any information related to the users, which is present in the system. For instance, as it has been seen in the Facebook privacy policies, the user cannot control any information about what she likes. The normal behaviour is to assign the same audience of the post she liked (clicking the ‘like’ button on the post). In order to express policies about it, we can leverage the structure of the propositions of $\mathcal{FPPF}_{\text{Facebook}}$ by using the attribute $like_{\eta}$. The privacy policy $[\![\neg S_{\{Ag \setminus friends(u)\}}\ u.like_{\eta}]\!]_u$ means that only u's friends can know what u liked.

Similar to retweet, in Facebook one can *share* a given post. Similarly to liking, sharing is available to the same audience as the post, but sharing entails the consequence of expanding the audience of the post. Specifically, all people included in the audience of posts of the user who is sharing will be added to the original audience of the re-shared post. In $\mathcal{FPPF}_{\text{Facebook}}$ we could prevent this by explicitly restricting the audience of our posts as we did in *Who can see my stuff?* or by writing (assuming Σ to be extended with the action *share*) $[\![\neg SP^{u}_{friends(u)} share]\!]_u$, where explicitly is stated who could share my posts but without limiting their audience.

We have seen in Lemma 2 how distributed knowledge could be used to make some inference on the knowledge of certain agents. Its use for defining privacy policies would allow social network users to control information which could be inferred by a group of agents. For instance, an agent $u \in Ag$ could define the policy $[\![K_i\ u.location \implies \neg D_{\{friends(u) \setminus \{i\}\}}\ u.location]\!]_u$ for a given agent $i \in friends(u)$, meaning that if one of u's friends already know u's location then the distributed knowledge between the rest of u's friends is not allowed. This example also exposes the usefulness of conditional privacy policies.

Interaction among several social networks

SN usually focusses on one particular kind of leisure. For instance, Twitter and Facebook both focus on sharing information among followers and friends, while others have a completely different focus, e.g. Spotify (music), Instagram (photos), and Youtube (videos). We have so far shown how to formalise single SN. We discuss in what follow some examples of privacy policies involving more than one social network.

For example, in Twitter it is possible to connect the account to a Facebook account. If a user enables it, she can choose to post her tweets and retweets on her Facebook timeline. The idea is that permissions should be set allowing or disallowing Twitter to post on a user's Facebook timeline. Due to the expressivity of $\mathcal{PPF}$ we can create an instantiation being the composition of Facebook and Twitter. For instance, if we combine $\mathcal{FPPF}_{\text{Twitter}}$ and $\mathcal{FPPF}_{\text{Facebook}}$, assuming a common set of agents Ag, and the union of the connections, auxiliary functions, actions, assumptions and restrictions over policies of both SN, we can write the following privacy policy: $[\![\neg S_{\{Ag \setminus (friends(u) \cap Followers(u)) \setminus \{u\}\}}\ u.location]\!]_u$. That is, only agents who are followers of u in Twitter, and friends in Facebook are allowed to know u's location. More complex properties of this kind could be formalised in $\mathcal{PPF}$.

II.5 Related Work

The approach we have followed in this paper has been to formally define privacy policies based on a variant of of *epistemic logic* [29], where it is possible to express the knowledge of *multi-agent systems* (MAS). One way to give semantics to the logic is to use *possible worlds semantics* (also known as *Kripke models*), where it is not explicitly represented what the agents know, but rather the *uncertainty* in their knowledge. This has the advantage of allowing to represent complex formulae about who knows what (including nesting of knowledge and other operators generalising the notion). Another way to give semantics to epistemic logic is to use *interpreted systems* which represents agent's knowledge as a set of runs (computational paths). Both ways of giving semantics come with advantages and disadvantages: Kripke models come with a heritage of fundamental techniques allowing to prove properties about the specification, while interpreted systems are quite intuitive to model MAS [55]. The common key in both approaches is modelling the uncertainty of the agent by using an equivalence relation. If one thinks about all the worlds that a given agent could consider possible in a social network system, it is easy to see that modelling them would lead to creating an enormous state space. Instead of modelling uncertainty we explicitly store what the agents know. This allows a more concise representation of the individuals' knowledge. Unlike previous work on epistemic logic, in our formalism worlds represent agents.

Moreover we explicitly model a restricted version of permission, i.e. our model explicitly shows which actions are allowed to be executed by the agents. Aucher *et al.* [3] show a different way of combining epistemic and deontic aspects in logic. They preserve the equivalence relation for epistemic properties and use an extra equivalence relation for representing permission. The logic is quite expressive but it suffers from the aforementioned state explosion problem. Furthermore the framework is defined as a mono agent system not being suitable for SN. We took their idea of combining epistemic and deontic operators in one language, but we restricted the semantic model according to the needs of SN.

In [84] Seligman *et al.* present a language based on epistemic logic, with the traditional Kripke semantics for the logic extended with a friendship relationship. By doing that they are able to reason about knowledge and friendship. Moreover they model a set of events using general dynamic dynamic logic (GDDL) by defining an *update* operation over the mentioned Kripke model. This enables the possibility of update the model as the events in the social network occur. Using GDDL they implement the concept of public and private announcement, which appear regularly in the communications among the agents. Although this approach is quite expressive, its focus is not on privacy or security issues, but in reasoning about the general knowledge of the agents. As mentioned before it comes with the price of having a immense state space and it complicates a practical implementation and the definition of an efficient (computationally speaking) model checking algorithm. Ruan and Thielscher [80] present a very similar formalism, but only public announcement is defined. Their focus is not on privacy either, but in the analysis of the "revolt or stay at home" effect, i.e. how the knowledge is spread among the agents.

There are other approaches for privacy not based on epistemic logic. One of the most interesting is Relationship-based access control (REBAC) [31]. The main difference with epistemic logic is that in REBAC the reasoning is focused on the resources own by the agents of the system. This approach is highly suitable for a practical implementation of a policy checking algorithm. On the other hand their approach would not detect certain kind of implicit knowledge flow. For instance, certain information about a user can be known after a friend of her is posting some information about both. The formalism is equipped with a formal language based on hybrid logic [12].

Datta *et al.* present in [20] the logic PrivacyLFP for defining privacy policies based on a restricted version of first-order logic (the restriction concerns that quantification over infinite values is avoided by considering only relevant instances of variables). The logic is quite expressive as it can represent things others than the kind of policies we are aiming at in this paper (their application domain being medical data). Though promising as a formalism for SN, the authors write that the logic might need to be adapted in order to be used for online social networks. To the best of our knowledge this has not been done.

II.6 Final Discussion

We have presented in this paper a framework for writing privacy policies for social networks. Our approach allows for the instantiation of the framework to formalise existing social networks, and other more complex privacy policies. One particularity of our approach is that worlds represent agents, closely following the structure of real social networks.

This paper is a first step towards a full formalisation of privacy policies for social networks. Our current and future work includes: **Adding real-time:** So far we cannot express policies with deadlines. This might be interesting in case policies are transient (e.g., "nobody is permitted to know my location during the first two weeks of May").

Modeling dynamic networks: The model we have of social networks is static, as well as the conformance relation between policies and the network. In practice the social network evolves, new friends come into place, others are blocked, etc. We aim at extending our formal model to capture such temporal aspect. **Adding roles and ontologies:** Agents in the SN could play different roles, e.g. individuals, companies, advertisement, etc. Providing $\mathcal{PPF}$ with the ability of detecting these roles would enhance its expressivity. **Developing an enforcing mechanism:** We have not mentioned how the policies might be enforced at runtime. We will explore how to extract a runtime monitor from the policy. Finally, we would like to explore the application of privacy-by-design [44] to a formalisation of social networks.

Chapter III

Formalising Privacy Policies in Social Networks

Raúl Pardo, Musard Balliu and Gerardo Schneider

Abstract. Social Network Services (SNS) have changed the way people communicate, bringing many benefits but also new concerns. Privacy is one of them. We present a framework to write privacy policies for SNSs and to reason about such policies in the presence of events making the network evolve. The framework includes a model of SNSs, a logic to specify properties and to reason about the knowledge of the users (agents) of the SNS, and a formal language to write privacy policies. Agents are enhanced with a reasoning engine allowing the inference of knowledge from previously acquired knowledge. To describe the way SNSs may evolve, we provide operational semantics rules which are classified into four categories: epistemic, topological, policy, and hybrid, depending on whether the events under consideration change the knowledge of the SNS' users, the structure of the social graph, the privacy policies, or a combination of the above, respectively. We provide specific rules for describing Twitter's behaviour, and prove that it is privacy-preserving (i.e., that privacy is preserved under every possible event of the system). We also show how Twitter and Facebook are not privacy-preserving in the presence of additional natural privacy policies.

III.1 Introduction

Over the past decade, the use of Social Network Services (SNS) like Facebook and Twitter has increased to the point of becoming ubiquitous. A recent survey shows that nearly 70% of the Internet users are active on SNSs [45]. Empirical studies show that the number of privacy breaches is keeping pace with this growth and users' requirements are much higher than the privacy guarantees offered by SNSs [6, 56, 41, 51, 57].

Motivation: In this paper we are concerned with privacy issues in SNSs. According to Boyd and Ellison [10] SNSs have three distinguishing characteristics that differentiate them from other services: i) A public profile; ii) A set of connections between users; iii) The ability for users to see certain information about others they are connected to, including meta-information such as others' connections. These features make SNSs susceptible to privacy breaches at different levels. Though the users of an SNS control much of the information disclosed about themselves, to date it is not clear whether such controls match the users' privacy intentions [26]. An additional concern is whether the privacy settings currently available in SNSs are suitable for capturing the needs of most users through a good privacy policy language. Privacy policies should also take into account that the networks *evolve*, for instance, when introducing new users or sending posts to other users. Many desirable privacy policies can already be enforced by SNSs; for instance, in Facebook users can state policies like "Only my friends can see posts on my timeline". Many other policies, however, are not possible to enforce, although they might be important from a user's privacy perspective. Again, in Facebook users can not specify privacy policies like "I do not want to be tagged in pictures by anyone other than myself" (P1) or "Nobody apart from myself can know my child's location" (P2). Although SNSs put more and more effort in improving users' privacy, the increasing amount of information that SNSs have to deal with and the continuous policy changes make this task cumbersome and hard to accomplish.

SNSs use different flavours of access control mechanisms to constrain the access to some piece of information and thus enforce the privacy policies. In particular, these mechanism would enforce policies like P1 and P2 by restricting the audience of the information or by defining the set of users that can perform some action, respectively. Unfortunately, these solutions come at the price of reducing the amount of information that can be shared in the SNS, hence making it less usable and attractive. Moreover, access control is not enough. Consider a Facebook user who sets the default audience of the pictures to her friends only. In Facebook, whenever a user is tagged on a picture, the audience of that picture is extended with the friends of the tagged user, hence the friend-only strategy can easily be infringed by tagging. Many other actions allow for implicit disclosure of knowledge; e.g. when joining an event users implicitly disclose their location to other participants of the event, or when commenting on a post the audience of the comment becomes the same as the audience of the post.

Our proposal: Our aim in this paper is to provide a suitable formalism for writing and reasoning about privacy policies in *dynamic* SNSs, and to enable a formal assessment on whether these policies are properly enforced by the SNSs. Our starting point is the definition of a formal framework for privacy policies consisting of: i) a generic model for social networks, ii) a knowledge-based logic to reason about the social network and privacy policies; iii) a formal language to describe privacy policies (based on the logic above).
Epistemic logic has been successfully used as a formal specification language in many settings [29, 75, 38]. Here, we advocate that this logic is natural for privacy in SNSs. We start with first-order logic to represent connections between users and enrich it with epistemic (K) and deontic (P) operators to express knowledge and permissions, respectively. For instance, if the logical formula $P_{Bob}^{Alice} tag$ specifies that "Bob is permitted to tag Alice in any picture", then we can write $[\![\neg P_{other}^{me} tag]\!]_{me}$ to model the policy P1 above. The wrapper $[\![\,]\!]_{me}$ is used to specify the owner of the privacy policy, in this case me. Similarly, if $S_{All}\, location(me)$ stands for "Someone among all users in the SNS knows my location", we can write $[\![\neg S_{All \setminus \{me\}} location(myChild)]\!]_{me}$ to specify the policy P2. Moreover, we can express more precise policies by nesting knowledge operators. Suppose Charlie is organising an event ev and he wants both Alice and Bob to participate; however, Alice will not participate if she knows that Bob is going. Then, Charlie, who definitively wants Alice to participate, can write the formula $[\![\neg K_{Alice} K_{Bob} ev]\!]_{Charlie}$ to express the policy "Alice can not know that Bob knows about the event ev".

We do not explicitly use the standard Kripke semantics to define satisfaction of a formula in our logic. Mainly, this is because we aim at providing a model that preserves the inherent structure of an SNS, whereas Kripke semantics requires a technical machinery that is far from a real model of social networks. Our framework directly captures the underlying structure of SNSs and thus brings out a faithful model of reality. On the other hand, we borrow traditional epistemic logic axiomatisations to define a *deductive engine* which determines the knowledge of the users. This enables us to, firstly, reuse existing theorem provers for checking whether a user knows some information, and secondly, choose weaker axiomatisations of knowledge that in turn are known to be more efficient to compute [29], thus paving the way for automated enforcement of privacy policies.

We extend the reasoning machinery with generic operational semantics rules which are used to model the dynamics of SNSs. The rules are divided in four categories depending on how the knowledge, the permissions, the social graph topology, the policy or a combination of them evolve as the users perform actions. We then give a full instantiation of our framework for Twitter and formally prove that it is privacy-preserving. We also consider desirable privacy policies that are currently not supported by SNSs, and show that Twitter is not privacy-preserving under those policies.

Contributions: More concretely, our contributions in this paper are:

1. An epistemic first-order framework $\mathcal{FPPF}$ for defining and reasoning about privacy policies (Section III.2), having the following features: i) A social network model (SNM), empowered with a deductive engine and closed under an axiom system for (first-order) epistemic logic, to generate implicit knowledge from existing knowledge in a knowledge base; ii) A first-order (relational) structure allowing for the modelling of rich relations and predicates; iii) A non-standard knowledge-based logic defined with the usual epistemic operators; iv) A formal language for defining expressive privacy policies, including nested knowledge.
2. A generic operational semantics for describing the behaviour of SNSs. The rules are of four different types (Section III.3.3): i) *Epistemic*, concerned with changes in the knowledge of a user; ii) *Topological*, concerned with changes in the structure of the network graph; iii) *Policy*, concerned with changes in the privacy policies; iv) *Hybrid*, a combination of the above three types of rules. We instantiate the generic semantics rules with rules for describing Twitter's behaviour (Section III.3.4).
3. A proof that Twitter is privacy-preserving with respect to all modelled events and privacy policies; a proof that Facebook is privacy-preserving with respect to a subset of events (Section III.4).
4. A proof that Twitter and Facebook are not privacy-preserving with respect to new desirable policies (Section III.4).

We are not the first to apply epistemic reasoning in the context of privacy for SNSs. A precursor of our approach is the framework proposed earlier by two of the authors in [71]. That framework only considers a *static* picture of SNSs and, besides the general template of $\mathcal{PPF}$ (see Definition 1), the two are fundamentally different. Our work redefines $\mathcal{PPF}$ entirely and introduces a novel approach to reason about the dynamic behaviour of SNSs. We use first-order structures enriched with the epistemic modality to increase the expressiveness of the logic and the policy language. We remark that the satisfaction relation uses a deductive engine over the user's knowledge base, which fixes unnecessary complications in the semantics given in [71]. We discuss the differences between our work and the work in [71] in more detail in Section VI.4.

III.2 Privacy Policy Framework

In this section we present a novel Privacy Policy Framework for social networks. As we will see, the framework is powerful enough to capture the features of today's social networks and at the same time it allows one to reason about privacy policy requirements of the users in a precise and formal manner. The framework is initially defined for generic social networks, however, not all SNSs have the same particularities. Due to this, we introduce the concept of *instantiation*, and show how to use it to instantiate Twitter.

The first-order privacy policy framework is equipped with several components. Firstly, we define models which leverage the well-known model for SNSs, the social graph [27]. We enrich these models with the knowledge and the permission that users have in an SNS. We represent the knowledge using a first-order epistemic (knowledge-based) structure, very much in the style of interpreted systems [29], and the permissions as links between users in the graph, similar to connections. Secondly, we introduce a knowledge-based logic to reason about the properties of the model. Finally, based on the logic, we provide an expressive language to write privacy policies. Formally, we define the framework as follows:

Definition 1 (First-Order Privacy Policy Framework)**.** *The tuple* $\langle \mathcal{SN}, \mathcal{KBL}, \vDash, \mathcal{PPL}, \vDash_C \rangle$ *is a* first-order privacy policy framework *(denoted by* $\mathcal{FPPF}$*), where*

- $\mathcal{SN}$ *is the set of all possible social network models;*
- $\mathcal{KBL}$ *is a knowledge-based logic;*
- $\vDash$ *is a satisfaction relation defined for* $\mathcal{KBL}$*;*
- $\mathcal{PPL}$ *is a formal language for writing privacy policies;*
- $\vDash_C$ *is a conformance relation defined for* $\mathcal{PPL}$*.* $\square$

High-level overview Figure III.1 provides an overview of the structure and relation between the components of the framework. The ultimate goal of our work is to define a privacy policy language, $\mathcal{PPL}$, for writing expressive privacy policies and checking their satisfaction for a social network model. To this end, we define a conformance relation, $\vDash_C$, that determines whether a privacy policy is in conformance with a given social network model. As shown in Figure III.1, the conformance relation relies on the satisfaction relation $\vDash$ of a more general logic, $\mathcal{KBL}$. We convert privacy policies to $\mathcal{KBL}$ formulae, since, as we will see, the syntax of $\mathcal{PPL}$ is a restricted form of the syntax of $\mathcal{KBL}$. We define the satisfaction relation $\vDash$ to check $\mathcal{KBL}$ formulae in a social network model, therefore reducing $\vDash_C$ to $\vDash$. Social network models consist of nodes that represent users and arrows that represent connections and permissions between users. We will describe social network models in full detail in the next section. In a

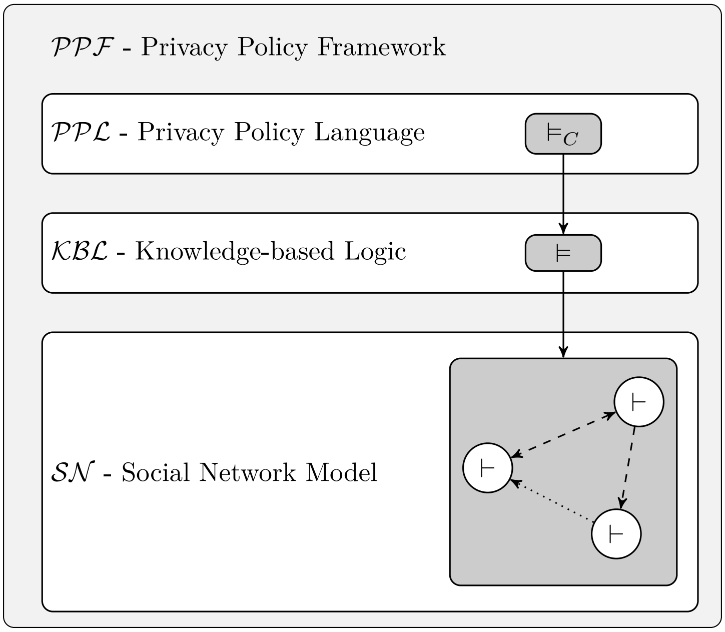

$\mathcal{PPF}$ - Privacy Policy Framework
$\mathcal{PPL}$ - Privacy Policy Language
$\models_C$
$\mathcal{KBL}$ - Knowledge-based Logic
$\models$
$\mathcal{SN}$ - Social Network Model
$\vdash$
$\vdash$
$\vdash$

Figure III.1: Structure of $\mathcal{PPF}$

nutshell, each node (user) i contains a local knowledge base and a derivability relation $\vdash$, which we use to determine whether the user i knows some information φ, namely the satisfaction of formulae $K_i\varphi$. We define this by requiring that φ is derivable (using $\vdash$) from the set of facts in the user's knowledge base. The derivability relation borrows the axioms and derivation rules from the **S5** axiomatisation defined by Fagin *et al.* in [29]. By using the **S5** axiomatisation for our notion of derivability (inside the users' knowledge bases), we can obtain an axiomatisation of the $\mathcal{KBL}$ logic that corresponds to the **KD45** axiomatisation and it is sound with respect to social network models [72]. In what follows we provide a detailed description of the components and their relations.

III.2.1 Social Network Models

Social networks are usually modelled as graphs, where nodes represent users—referred to as *agents*—and edges represent different kinds of relationships among users, for instance, friendship or family. These graphs are traditionally called social graphs [27]. A social network model is a social graph which includes the knowledge that the agents have accumulated as a set of logic formulae in their *knowledge base*. Moreover, we model possible inferences of knowledge that the agents can make from the knowledge that they already possess. Additionally, we use new types of edges to represent certain permission that the agents may have. We write $\mathcal{AU}$ to denote a universe of agents.

Definition 2. *Given a set of formulae $\mathcal{F}$, a set of privacy policies Π, and a finite set of agents $Ag \subseteq \mathcal{AU}$, a* social network model *(SNM) is a social graph of the form $\langle Ag, \mathcal{A}, KB, \pi \rangle$, where*

- *Ag is a nonempty finite set of* nodes *representing the agents in the SNS.*
- *$\mathcal{A}$ is a first-order structure over the social network model. As usual, it consists of a set of domains; and a set of relations, functions and constants interpreted over their corresponding domain.*
- *$KB : Ag \to 2^{\mathcal{F}}$ is a function denoting the* knowledge base *of an agent, namely the accumulated knowledge of an agent. We write KB_i to denote $KB(i)$.*
- *$\pi : Ag \to 2^{\Pi}$ is a function specifying the set of privacy policies of each agent. We write π_i for $\pi(i)$.* □

In Definition 2, the shape of the relational structure $\mathcal{A}$ depends on the type of the social network under consideration. We represent the connections and the permission actions between social network agents, i.e., edges of the social graph, as families of binary relations, respectively $\{C_i\}_{i \in \mathcal{C}} \subseteq Ag \times Ag$ and $\{A_i\}_{i \in \Sigma} \subseteq Ag \times Ag$ over the domain of agents. We use $\{D_i\}_{i \in \mathcal{D}}$ to denote the set of domains. The set of agents Ag is always included in the set of domains. We use $\mathcal{C}, \Sigma$ and $\mathcal{D}$ to denote sets of indexes for connections, permissions and domains, respectively. Sometimes, we use predicates, e.g. *friends*(A, B), to denote that the elements $A, B \in Ag$ belong to the binary relation *friends* defined over pairs of agents as expected.

We provide agents with reasoning capabilities that allow them to infer new knowledge. For instance, in Facebook, events include the location where the event takes place. Imagine that Alice knows that Bob is attending an event. Given that the event information includes the location, she must also know Bob's location. The reasoning capabilities for the agents help us to make these type of inferences in the framework.

Since the knowledge of the agents is represented using epistemic logic formulae, we use standard properties of knowledge to model the reasoning capabilities of agents. We introduce a set of axioms and rules for an agent to infer new knowledge from the one present in their knowledge base. In particular, we use the knowledge axiomatisation **S5** from first-order epistemic logic [29, 59]. **S5** is a standard and widely used axiomatisation for epistemic logic. Moreover, there exist several tools to check validity of epistemic formulae for **S5**. These tools can be used to implement an enforcement mechanism based on our framework. Nevertheless, we are not restricted to **S5**. Given the modularity of our approach, we could consider other axiomatisations.

We now define the language for first-order epistemic logic, $\mathcal{L}_n$:

$$\varphi \quad ::= \quad p(\overrightarrow{t}) \mid \varphi \wedge \varphi \mid \neg\varphi \mid \forall x.\varphi \mid K_i\varphi.$$

where i is an agent from a set of agents Ag and $p(\overrightarrow{t})$ is a predicate over terms $\overrightarrow{t}$. For now it is enough to assume that a *term* is either a constant symbol, a function symbol (with implicit arity), or a variable. The intuitive reading for the formula $K_i\varphi$ is "agent i knows φ". Hence we can use predicates to denote concrete pieces of information, e.g. Bob's location can be written as $location(Bob)$ and the statement "Alice knows Bob's location" can be written as $K_{Alice}location(Bob)$.

Now we introduce the set of properties of knowledge as defined by the **S5** *axiomatisation.*

Definition 3 (First-Order **S5** [29, 59]). *Given the formulae φ and ψ written in $\mathcal{L}_n$ and some agent i, the axiom system **S5** consists of the following axioms and derivation rules:*

Axioms

(A1) All (instances of) first-order tautologies

(A2) $(K_i\varphi \wedge K_i(\varphi \implies \psi)) \implies K_i\psi$

(A12) $\forall x_1, \cdots, x_k.K_i\varphi \implies K_i\forall x_1, \cdots, x_k.\varphi$

(A3) $K_i\varphi \implies \varphi$

(A4) $K_i\varphi \implies K_iK_i\varphi$

(A5) $\neg K_i\varphi \implies K_i\neg K_i\varphi$

Derivation rules

(Modus Ponens) $\dfrac{\varphi \qquad \varphi \implies \psi}{\psi}$

(Necessitation) $\dfrac{\varphi}{K_i\varphi}$ *where φ must be provable from no assumptions.*

(Generalisation) $\dfrac{\varphi}{\forall x.\varphi(x)}$ *where x does not occur in φ.*

We also introduce the notion of derivation in the axiom system **S5** as follows:

Definition 4 ([59]). *A* derivation *of a formula $\varphi \in \mathcal{L}_n$ is a finite sequence of formulae $\varphi_1, \varphi_2, \ldots, \varphi_n = \varphi$, where each φ_i, for $1 \leq i \leq n$, is either an instance of the axioms (A1, A2, A12, A3, A4 or A5) or the conclusion of one of the derivation rules of which premises have already been derived, i.e., appear as φ_j with $j < i$. Moreover when we can derive φ from a set of formulae $\{\psi_1, \psi_2, \ldots \psi_n\}$, if we take the set Γ as the conjunction of all the formulae from the previous set, $\Gamma = \psi_1 \land \psi_2 \land \ldots \land \psi_n$ we write $\Gamma \vdash \varphi$.*

Consider again the previous example about Alice and Bob. We can formalise the statement "Alice knows that if a user is going to an event, then she knows the location of that user during the event" as

$$K_{Alice}(going(u, \eta) \implies location(u, \eta)) \tag{III.1}$$

for each $u \in Ag$ and $\eta \in \mathbb{N}$. Hence if Alice knows that Bob is going to the event η,

$$K_{Alice}going(Bob, \eta) \tag{III.2}$$

she can apply the axiom (A2) together with (III.1) and (VI.2) to infer Bob's location during the event, i.e., $K_{Alice}location(Bob, \eta)$.

Finally, we introduce the closure function *Cl* for the axiom system **S5**, which generates all the knowledge that agents can infer given the set formulae representing the explicit knowledge that they already have. This is formally defined as follows:

Definition 5. *Given a set of formulae $\Phi \subseteq \mathcal{L}_n$ the* knowledge base *closure function is $Cl(\Phi) = \{\varphi \mid \Phi \vdash \varphi\}$.*

Cl is the closure of an input set of formulae under the axiom system **S5**. Different axioms may hold when $\mathcal{FPPF}$ is instantiated with a concrete social network model. To this end we define the minimal *Cl* using the axioms and derivation rules from **S5** and, in order to provide the agents with more targeted deductive engines, we remain open to extend this set of axioms. We ensure that the local knowledge of each agent is always consistent by checking that false is never derived. In Section III.2.4 we will show how *Cl* can be extended to meet the requirements of a concrete SNS.

Example 1. *Consider an $SN \in \mathcal{SN}$ which consists of three agents Alice, Bob and Charlie, $Ag = \{Alice, Bob, Charlie\}$; two connections Friendship and Blocked, $\mathcal{C} = \{Friendship, Blocked\}$; and a friend request action, $\Sigma = \{friendRequest\}$.*

Figure III.2 shows a graphical representation of the aforementioned SN. In this model the dashed arrows represent connections. Note that the Friendship connection

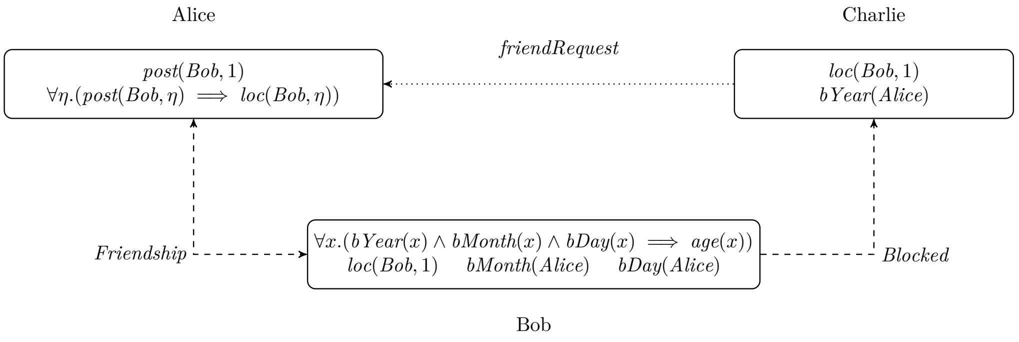

Alice
Charlie
friendRequest
$post(Bob,1)$
$\forall\eta.(post(Bob,\eta) \implies loc(Bob,\eta))$
$loc(Bob,1)$
$bYear(Alice)$
Friendship
$\forall x.(bYear(x) \wedge bMonth(x) \wedge bDay(x) \implies age(x))$
$loc(Bob,1)$ $bMonth(Alice)$ $bDay(Alice)$
Blocked
Bob

Figure III.2: Example of Social Network Model

is bidirectional, i.e., Alice is friend with Bob and vice versa. On the other hand, it is also possible to represent unidirectional connections, as Blocked; in SN Bob has blocked Charlie. Permissions are represented using a dotted arrow. In this example, Charlie is able to send a friend request to Alice.

The predicates inside each node represent the agents' knowledge base. In this SNM, Charlie and Bob have the predicate $loc(Bob, 1)$[1] in their knowledge bases, meaning that they both know location number 1 of Bob. Moreover, the knowledge bases may contain not only predicates, but also other formulae. These formulae may increase the knowledge of the agents. For instance, Alice knows $loc(Bob, 1)$ implicitly. Alice can derive $loc(Bob, 1)$ by Modus Ponens*, from $post(Bob, 1)$ and $\forall\eta.post(Bob, \eta) \implies loc(Bob, \eta)$.* $\square$

III.2.2 Knowledge-based Logic

We use the logic $\mathcal{KBL}$ to reason about the knowledge and the permissions of agents over social network models. The logic allows us to leverage all the expressive power of first-order epistemic reasoning to formally express and verify privacy policies. As usual in first-order logic, we start with a *vocabulary* consisting of a set of constant symbols, variables, function symbols and predicate symbols, which are used to define terms as follows:

Definition 6 (Terms)**.** *Let x be a variable and c a constant and $\{f_i\}$ for $i \in \mathcal{I}$ (where $\mathcal{I}$ is a set of indexes) a family of functions with implicit arity. Then the* terms *are*

[1]Since a user can have several locations we use an index to differentiate them, $loc(Bob, 1)$ represents location 1 of Bob.

inductively defined as:

$$t \quad ::= \quad c \mid x \mid f_i(\overrightarrow{t})$$

where $\overrightarrow{t}$ denotes a tuple of terms respecting the arity of f_i.

We use terms to define predicates. For instance, *friends*$(Alice, Bob)$ is a predicate that can be used to express that Alice and Bob are friends. The syntax of the logic is then defined as follows:

Definition 7 (Syntax)**.** *Given $i, j \in Ag$, the relation symbols $a_n(i,j)$, $c_m(i,j)$, $p(\overrightarrow{t}) \in \mathcal{A}$ where $m \in \mathcal{C}$ and $n \in \Sigma$, and a nonempty set $G \subseteq Ag$, the syntax of the* knowledge-based logic *$\mathcal{KBL}$ is inductively defined as:*

$$\varphi \quad ::= \quad p(\overrightarrow{t}) \mid c_m(i,j) \mid a_n(i,j) \mid \varphi \wedge \varphi \mid \neg\varphi \mid \forall x.\varphi \mid K_i\varphi \mid D_G\varphi \mid C_G\varphi$$

The remaining epistemic modalities are defined as $S_G\varphi \triangleq \bigvee_{i \in G} K_i\varphi$ and $E_G\varphi \triangleq \bigwedge_{i \in G} K_i\varphi$.

We choose to discriminate between predicates encoding permissions between agents, i.e., $a_n(i,j)$, predicates encoding connections between agents, i.e., $c_m(i,j)$, and other types of predicates, e.g. $p(\overrightarrow{t})$, in order to stay as close as possible to the social network models. $\mathcal{F}_{\mathcal{KBL}}$ will represent the set of all well-formed formulae of $\mathcal{KBL}$ according to the category φ above. The epistemic modalities stand for: $K_i\varphi$, agent i knows φ; $E_G\varphi$, everyone in the group G knows φ; $S_G\varphi$, someone in the group G knows φ; $D_G\varphi$, φ is distributed knowledge in the group G; $C_G\varphi$, φ is common knowledge in the group G. We use the following operators as syntactic sugar in the logic $\mathcal{KBL}$: $P_i^j a_n := a_n(i,j)$, agent i is permitted to execute action a_n to agent j; $SP_G^j a_n := \bigvee_{i \in G} a_n(i,j)$, at least one agent in G is permitted to execute action a to agent j; $GP_G^j a_n := \bigwedge_{i \in G} a_n(i,j)$, all agents in G are permitted to execute action a to agent j. When we write "*agent i is permitted to execute action a_n to agent j*", it means that agent i allows j to perform an action a_n which directly involves i, e.g. P_{Bob}^{Alice}*friendRequest* would mean that Bob is allowed to send a friend request to Alice. We define E_G^{k+1} as $E_G E_G^k \varphi$, where $E_G^0\varphi$ is equal to φ. The logical operators $\rightarrow$ and $\vee$ are defined in terms of $\wedge$ and $\neg$ as usual.

In what follows we define the satisfaction relation for $\mathcal{KBL}$ formulae, interpreted over social network models.

Definition 8. *Given a social network model $SN = \langle Ag, \mathcal{A}, KB, \pi \rangle$, the agents $i, j, u \in Ag$, $\varphi, \psi \in \mathcal{F}_{\mathcal{KBL}}$, a nonempty set of agents $G \subseteq Ag$, $m \in \mathcal{C}$, $n \in \Sigma$, $o \in \mathcal{D}$ and $k \in \mathbb{N}$, the* satisfaction relation *$\vDash \subseteq \mathcal{SN} \times \mathcal{KBL}$ is defined as shown in Table III.1.* $\square$

$SN \models p(\vec{t})$	iff	$p(\vec{t}) \in KB_e$
$SN \models c_m(i,j)$	iff	$(i,j) \in C_m$
$SN \models a_n(i,j)$	iff	$(i,j) \in A_n$
$SN \models \neg\varphi$	iff	$SN \not\models \varphi$
$SN \models \varphi \wedge \psi$	iff	$SN \models \varphi$ and $SN \models \psi$
$SN \models \forall x.\varphi$	iff	for all $v \in D_o$, $SN \models \varphi[v/x]$
$SN \models K_i\varphi$	iff	$\varphi \in Cl(KB_i)$
$SN \models C_G\varphi$	iff	$SN \models E_G^k\varphi$ for $k = 1, 2, \ldots$
$SN \models D_G\varphi$	iff	$\varphi \in Cl(\bigcup_{i \in G} KB_i)$

Table III.1: $\mathcal{KBL}$ satisfaction relation

Predicates are interpreted as relations over the respective domains in the model, e.g., connections and permissions. Additionally, we introduce a special agent e, called environment, that defines the truth of atomic formulae of the type $p(\vec{t})$. The environment's knowledge base (KB_e) contains all predicates $p(\vec{t})$ that are true in the real world, but the other agents may not know about. Intuitively, the environment agent can be seen as an oracle that knows all the true facts in the real world, for instance, $Alice \neq Bob$ or $Alice \notin \{Bob, Charlie\}$. For simplicity, sometimes we use predicates to represent pieces of information. These predicates are always present in KB_e. For instance, in Example 1, we use the predicate $loc(Bob, 1)$ to represent location 1 of Bob, which allows us to write formulae such as $K_{Alice}loc(Bob, 1)$ to state that "Alice knows location 1 of Bob". As usual, $\varphi[v/x]$ denotes the capture-free substitution in first-order logic and we tacitly assume that each variable x is mapped to its own domain.

The intuition behind the semantic definition of the knowledge modality K_i is as follows: a user i knows φ (denoted as $K_i\varphi$) iff either the user knows φ explicitly, i.e., φ is in the knowledge base (KB_i), or φ can be derived (using the axiomatisation **S5**) from the existing formulae in the knowledge base ($\varphi \in Cl(KB_i)$). This definition is better illustrated by an example.

Example 2. *Let SN be the SNM in Figure III.2. As we described in Example 1, Alice knows post 1 of Bob, meaning that*

$$SN \models K_{Alice}post(Bob, 1)$$

holds, since $post(Bob, 1)$ is explicitly in the knowledge base of Alice, i.e.

$$post(Bob, 1) \in KB_{Alice}. \tag{III.3}$$

We also mentioned that Alice implicitly knows location 1 of Bob, which means that

$$SN \vDash K_{Alice} loc(Bob, 1) \tag{III.4}$$

should hold. According to the semantics we have provided for K_i, the previous statement is true iff $loc(Bob, 1) \in Cl(KB_{Alice})$. Figure III.2 shows that, in SN, the following formula is in KB_{Alice}

$$\forall \eta . post(Bob, \eta) \implies loc(Bob, \eta) \tag{III.5}$$

where $\eta \in \mathbb{N}$, hence

$$post(Bob, 1) \implies loc(Bob, 1) \tag{III.6}$$

is also in KB_{Alice}. From (III.3) and (III.6) we know that the knowledge base of Alice contains at least the following elements,

$$KB_{Alice} = \{post(Bob, 1), post(Bob, 1) \implies loc(Bob, 1), \ldots\}.$$

Finally, by the definition of Cl (Definition 9), modus ponens *can by applied for (III.6) and (III.3) to derive $loc(Bob, 1)$, i.e. $loc(Bob, 1) \in Cl(KB_{Alice})$ and therefore (III.4) holds.* □

The interpretation of distributed knowledge, D_G, is similar to the one for K_i, but it considers the knowledge of all agents in G instead of accounting only for the knowledge of agent i.

We can use the logic $\mathcal{KBL}$ to reason about combinations of what the agents know and what actions they are allowed to perform in an SNM.

Example 3. *Consider again the SNM SN in Figure III.2. We can check whether the statement*

$$SN \vDash E_{\{Bob, Charlie\}} loc(Bob, 1) \implies P^{Alice}_{Charlie} friendRequest$$

holds for $i \in \{Alice, Bob, Charlie\}$. As in Example 1, Bob and Charlie both know location 1 of Bob, therefore it holds that $loc(Bob, 1) \in Cl(KB_{Bob})$ and $loc(Bob, 1) \in Cl(KB_{Charlie})$. Hence

$$SN \vDash K_{Bob} loc(Bob, 1) \wedge K_{Charlie} loc(Bob, 1),$$

it implies that

$$SN \vDash E_{\{Bob, Charlie\}} loc(Bob, 1).$$

Also $(Charlie, Alice) \in A_{friendRequest}$, meaning that Charlie is permitted to send a friend *request to Alice, therefore it holds*

$$SN \vDash P^{Alice}_{Charlie} friendRequest.$$

Finally we can conclude that our original implication holds for SN. □

Not all SNSs have the same knowledge and permission properties. Different properties hold in different SNSs. As we have seen, using the satisfaction relation $\vDash$, we can determine whether a $\mathcal{KBL}$ formula holds in an SNM. These knowledge and permission properties can be expressed in $\mathcal{KBL}$, and consequently, we can check whether they hold in a specific SNM as we show in the following example.

Example 4. *In Facebook, as soon as a user has access to a post, she can see all the users who liked the post. This means that when any member clicks the "like" button, all the users with access to the post will know about it. Let o be some agent and η some post, the predicate $post(o, \eta)$ representing the post η by agent o and the predicate $like(i, o, \eta)$ representing the fact that agent i liked the post η by o, we can check whether the property holds in a given $SN \in \mathcal{SN}$ using the satisfaction relation:*

$$SN \vDash \forall j.\forall o.\forall i.\forall \eta. K_j post(o, \eta) \wedge K_i like(i, o, \eta) \implies K_j like(i, o, \eta)$$

□

Properties of the framework. The logic $\mathcal{KBL}$ leverages the deductive engine by applying the axioms and derivation rules from the axiomatisation **S5** to infer new knowledge. We remark that the same axiomatisation can not be used for $\mathcal{KBL}$, since the predicates, e.g. connection and permission, are interpreted differently depending on whether or not they occur inside a knowledge modality. For instance, satisfaction of the connection predicate $friendship(Alice, Bob)$ will only depend on the condition $(Alice, Bob) \in A_{Friendship}$. Nonetheless, checking the formula $K_{Alice} friendship(Alice, Bob)$ requires that $friendship(Alice, Bob) \in Cl(KB_{Alice})$. This is in line with the fact that agents may know facts that are not true in the real world. As a result, this unconventional interpretation of predicates prevents us from using axiomatisations defined for classical epistemic logic. However, when checking if a formula is in the knowledge base of an agent, all predicates are treated equally, even when they are connection or permission predicates. Hence the individual knowledge of each agent in SNMs can be modelled using a classical Kripke model, meaning that it can be seen as a set of formulae in $\mathcal{L}_n$. Because of this, we assume that they can infer new knowledge using the axiomatisation **S5**, which is sound and complete for classical epistemic logics [29].

The example above shows that we are concern with agents' belief instead of knowledge. In fact, we can use the **KD45** axiomatisation for belief [29] to check properties of the logic $\mathcal{KBL}$ [72]. Intuitively, we can think of SNMs as models that combine two logics. On the one hand, we use $\mathcal{KBL}$ to reason about the global knowledge and permission of the SNS. On the other hand, agents can have their local knowledge represented using $\mathcal{L}_n$ and use **S5** to infer new knowledge. The $\mathcal{KBL}$ logic is closely related to traditional Kripke models as discussed in Section VI.4.

Finally, the logic relies on the assumption that the knowledge bases of the agents are always *consistent* in the sense that falsehood is never derived. In practice this assumption can be relaxed either by checking that false is never derived whenever adding new facts to a knowledge base, or by extending predicates with indexes/timestamps the discriminate between predicates added at different stages to a knowledge base.

III.2.3 The Privacy Policy Language

One of the objectives of $\mathcal{FPPF}$ is to provide a way to express complex and fine-grained privacy policies. We introduce $\mathcal{PPL}$ as a formal language for writing privacy policies based on $\mathcal{KBL}$.

Definition 9. *Given the agents $i, j \in Ag$, the relation symbols $a_n(i,j)$, $c_m(i,j)$, $p(\vec{t}) \in \mathcal{A}$ where $m \in \mathcal{C}$ and $n \in \Sigma$, a nonempty set $G \subseteq Ag$ and $\varphi \in \mathcal{F}_{\mathcal{KBL}}$, the syntax of the* privacy policy language *$\mathcal{PPL}$ is inductively defined as follows:*

$$\begin{array}{rcl}
\delta & ::= & \delta \wedge \delta \mid \forall x.\delta \mid [\![\varphi \implies \neg\alpha]\!]_i \mid [\![\neg\alpha]\!]_i \\
\alpha & ::= & \alpha \wedge \alpha \mid \psi \mid \gamma' \mid \forall x.\alpha \\
\gamma' & ::= & K_i\gamma \mid E_G\gamma \mid S_G\gamma \mid D_G\gamma \mid C_G\gamma \\
\gamma & ::= & \gamma \wedge \gamma \mid \neg\gamma \mid p(\vec{t}) \mid \gamma' \mid \psi \mid \forall x.\gamma \\
\psi & ::= & c_m(i,j) \mid a_n(i,j)
\end{array}$$

In $\mathcal{PPL}$ privacy policies are written in a negative way in order to specify who is not allowed to know a fact or who is not permitted to perform an action. Note that in δ, α is always preceded by negation. The syntactic category α represents the restrictions that must be enforced in the social network; the set of well-formed formulae of this category is denoted as $\mathcal{F}^{\mathcal{R}}_{\mathcal{PPL}}$. The category γ' corresponds to a restricted version of $\mathcal{F}_{\mathcal{KBL}}$ where the first element is a positive knowledge modality. This forces policies to be written in a negative way, since no double negation is possible in the first knowledge modality. Also, we always refer to the agents' knowledge, since γ' starts with a knowledge modality. The category ψ gives a special treatment of predicates for actions and connections to express restrictions over the connections and the actions that agents are involved in. In δ we wrap privacy policies using $[\![\ ]\!]_i$, where $i \in Ag$, to denote the owner of the privacy policy. We write $\mathcal{F}_{\mathcal{PPL}}$ for the set of well-formed $\mathcal{PPL}$ formulae given by δ. As a result, there are two main types of privacy policies that users can write:

- *Direct restrictions* - $[\![\neg\alpha]\!]_i$ These are restrictions which allow users to explicitly specify the audience which has no access to some piece of information or who is permitted to execute an action. For instance, in $\mathcal{PPL}$ agent i can write $[\![\neg S_{\{m,n,o\}} p(\vec{t})]\!]_i$, meaning that none of the agents $m, n, o \in Ag$ can know $p(\vec{t})$.
- *Conditional restrictions* - $[\![\varphi \implies \neg\alpha]\!]_i$ A restriction α is enforced depending on some knowledge or permission state (see Example 5).

Example 5. *Let us consider the following policy:*

$$\forall j.[\![\neg P^i_j join_{event(i)} \implies \neg K_j\, event(i, descp)]\!]_i \tag{III.7}$$

The intuitive meaning of this policy is that if a user $i \in Ag$ creates an event $event(i, descp)$ (where descp is the description of the event) and she gives permission to join it (the action of joining the event is represented by $join_{event(i)}$) to a certain group of people, then the event cannot be accessed by people other than the ones who are allowed to join it. Similarly, a user can choose to limit the event's audience to her friends only. This can be expressed in $\mathcal{PPL}$ as

$$[\![\neg S_{Ag\setminus friends(i)}\, event(i, descp)]\!]_i \tag{III.8}$$

Unlike (III.7), this policy is enforced in most SNSs. However (III.8) is much more coarse-grained than (III.7) and, as a result, it will not allow some users to access the event if they are able to join it. Therefore, (III.8) unnecessarily reduces the audience of the event.

We now give an example of a privacy policy which uses the distributed knowledge modality.

Example 6. *The distributed knowledge operator D_G makes possible to protect users' against intricate leaks of information in groups of agents. Consider the social network model presented in Figure III.2, where Bob knows the day and the month of Alice's birthday, denoted by $bDay(Alice)$ and $bMonth(Alice)$, respectively, and he can also infer the age of a user whenever he knows the user's full date of birth, i.e.,*

$$\forall x.bDay(x) \wedge bMonth(x) \wedge bYear(x) \implies age(x).$$

$SN \vDash_C \delta_1 \wedge \delta_2$	iff	$SN \vDash_C \delta_1 \wedge SN \vDash_C \delta_2$
$SN \vDash_C \forall x.\delta$	iff	for all $v \in D_o$, $SN \vDash_C \delta[v/x]$
$SN \vDash_C \llbracket \neg\alpha \rrbracket_i$	iff	$SN \vDash \neg\alpha$
$SN \vDash_C \llbracket \varphi \implies \neg\alpha \rrbracket_i$	iff	$SN \vDash \varphi$ then $SN \vDash_C \llbracket \neg\alpha \rrbracket_i$

Table III.2: $\mathcal{PPL}$ conformance relation

Moreover, Charlie knows the year of Alice's birth, represented by the predicate $bYear(Alice)$. Therefore, if Bob and Charlie combine their knowledge, Alice's age will become distributed knowledge between the two. This is because the distributed knowledge operator considers the combination of the knowledge of the group of agents and applies the deductive engine to infer new knowledge. Fortunately, in $\mathcal{PPL}$ Alice can write the privacy policy

$$\llbracket \neg D_{\{Bob,Charlie\}}\, age(Alice) \rrbracket_{Alice}$$

to prevent this leak. Note that the social network model considered in this example (Figure III.2) violates the policy. □

The examples above show that the privacy policies we express in $\mathcal{PPL}$ give users a fine-grained control over *what* information they share and with *whom* they share it. In order to ensure that users' privacy is not compromised, all their privacy policies must be in conformance with the SNS.

Definition 10. *Given a social network model $SN = \langle Ag, \mathcal{A}, KB, \pi \rangle$, an agent $i \in Ag$, $\varphi \in \mathcal{F}_{\mathcal{KBL}}$, $\alpha \in \mathcal{F}^{\mathcal{R}}_{\mathcal{PPL}}$, $o \in \mathcal{D}$ and $\delta, \delta_1, \delta_2 \in \mathcal{F}_{\mathcal{PPL}}$, the* conformance relation $\vDash_C$ *is defined as shown in Table III.2.* □

Note that $\vDash_C$ is defined using the satisfaction relation $\vDash$. Due to this, privacy policies can be seen as specific knowledge and permission conditions that must hold in the SNM. Let us take as an example the policy (III.7) from Example 5 and an SNM SN

$$SN \vDash_C \forall j. \llbracket \neg P^i_j join_{event(i)} \implies \neg K_j\, event(i, descp) \rrbracket_i .$$

By applying the semantics defined in Table III.2, checking whether SN is in conformance with the policy is equivalent to checking that for all $u \in Ag$

$$SN \vDash_C \llbracket \neg P^i_u join_{event(i)} \implies \neg K_u\, event(i, descp) \rrbracket_i$$

which is also equivalent to

$$\text{If } SN \vDash \neg P^i_u join_{event(i)} \text{ then } SN \vDash \neg K_u\, event(i, descp)$$

As we can see in Table III.2, checking conformance of any formula in $\mathcal{PPL}$ boils down to checking satisfaction of the corresponding formula in $\mathcal{KBL}$.

III.2.4 Instantiation of the framework

So far we have described a generic framework applicable to general SNSs. However, each SNS has its own features. For example, Foursquare has the follower connection and users can write tips related to places where they have been. In Google+ users are grouped in *circles* and they share information depending on those circles. Moreover, Google+ offers users the possibility of creating events that other users can join, whereas this is not present in other SNSs like Foursquare, Twitter or Instagram.

Here we introduce the concept of $\mathcal{FPPF}$ instantiation, which will be used to model the specific characteristics of an SNS.

Definition 11 ($\mathcal{FPPF}$ instantiation)**.** *An $\mathcal{FPPF}$* instantiation *for an SNS $\mathcal{S}$ is a tuple of the form*

$$\mathcal{FPPF}_{\mathcal{S}} = \langle \mathcal{SN}_{\mathcal{S}}, \mathcal{KBL}, \vDash, \mathcal{PPL}, \vDash_C \rangle$$

where $\mathcal{SN}_{\mathcal{S}} = \langle Ag_{\mathcal{S}}, \mathcal{A}_{\mathcal{S}}, KB_{\mathcal{S}}, \pi_{\mathcal{S}} \rangle$ and

- *$Ag_{\mathcal{S}}$ is the set of agents in the SNS;*
- *The structure $\mathcal{A}_{\mathcal{S}}$ contains a set of predicates $\mathcal{P}_{\mathcal{S}}$, a set of connection relations $\mathcal{C}_{\mathcal{S}}$, a set of permission relations $A_{\mathcal{S}}$, and a family of auxiliary functions $\{f_i\}_{i \in \mathcal{I}}$;*
- *The knowledge base contains a set of properties $\mathbb{A}_{\mathcal{S}}$ of the SNS, written in $\mathcal{KBL}$ (these properties represent assumptions for the instantiated SNS);*
- *$\pi_{\mathcal{S}}$ returns the set of privacy policies of an agent in $\mathcal{S}$. We assume that the set of privacy policies $\Pi_{\mathcal{S}}$ is consistent. We also assume that all privacy policies in $\Pi_{\mathcal{S}}$ satisfy the* admissibility *condition $\mathbb{RE}_{\mathcal{S}}$.*

The *admissibility* condition $\mathbb{RE}_{\mathcal{S}}$ specifies the *generic structure* of privacy policies for a particular instantiation (see Definition 12 for an example). Formally, $\mathbb{RE}_{\mathcal{S}}$ is a predicate over the elements of $\mathcal{FPPF}_{\mathcal{S}}$ defining the well-formed policies for the instantiation. We write $\pi' \in \mathbb{RE}_{\mathcal{S}}$ if π' satisfies $\mathbb{RE}_{\mathcal{S}}$. Independently of the admissibility condition, we assume that all privacy policies are consistent. For simplicity, when no confusion arises, we will not specify the subindex $\mathcal{S}$ in the instantiation. Also, as mentioned before, the deductive engine of the *knowledge base*, *KB*, is extended with the assumptions $\mathbb{A}_{\mathcal{S}}$ in the instantiation.

III.2.5 Instantiation of Twitter

Twitter is an SNS in which the interaction among users is carried out by means of posting (or tweeting) 140 characters messages called *tweets*. Apart from text, tweets can also include pictures and locations. If users want to re-share a tweet, they can use the *retweet* functionality which shares an already published tweet from another user. Users can also mark tweets as *favourite*, which is similar to star emails, i.e., it marks the tweet with a star. It has recently become quite trendy to use the favourite feature as a way to express that you like the content of the tweet. The main relationship between users is called *follower*. It is a unidirectional relation between users. When users follow another user, they get updates with all the tweets of the user they follow.

In what follows we formally present the Twitter instantiation, denoted by $\mathcal{FPPF}_{\text{Twitter}}$.

Predicates. Given $o, u \in Ag$ and $\mu, \eta \in \mathbb{N}$, the set of predicates $\mathcal{P}_{\text{Twitter}} \in \mathcal{FPPF}_{\text{Twitter}}$ is:

- $tweet(o, \eta)$ - Tweet η tweeted by o.
- $mention(u, o, \eta)$ - Mention of u in $tweet(o, \eta)$.
- $favourite(u, o, \eta)$ - u marked $tweet(o, \eta)$ as favourite.
- $retweet(u, o, \eta)$ - Retweet of $tweet(o, \eta)$ by u.
- $location(o, \eta)$ - Location of $tweet(o, \eta)$
- $picture(o, \eta, \mu)$ - A picture included in $tweet(o, \eta)$.
- $username(u), email(u), phone(u)$ - Username, email address and phone number of user u.

The constants η and μ are indexes for tweets and pictures of a user, respectively.

Connections. The set of connections include the follower and the block relationships, $\mathcal{C}_{\text{Twitter}} = \{C_{Follower}, C_{Block}\}$.

Actions. The actions are defined as:

$$A_{\text{Twitter}} = \{A_{accessProf}, A_{accessProfRec}, A_{sendAd}\}$$

where *accessProf* is the action of a user accessing other user's profile; the action *accessProfRec* represents a user's profile can be accessed as a recommendation, for example when a user installs the Twitter mobile app, the SNS recommends other users which may be in the user's contact list; and *sendAd* is the action of an advertisement company sending advertisements to a user.

Constraints over privacy policies (Admissibility condition). In $\mathcal{FPPF}_{\text{Twitter}}$ we do not define constraints for the privacy policies *per se*. Instead we describe a schema composed by the generic structure of the privacy policies that users in Twitter can write. The schema is based on the set of Twitter privacy policies presented in [71], which was shown to express all the policies that Twitter offers in its privacy settings section nowadays.

Definition 12. *Given $u \in Ag$ and $\eta \in \mathbb{N}$; the generic structure of the privacy policies for Twitter is as follows:*
$P1(u) = [\![\neg S_{Ag \setminus followers(u) \setminus \{u\}}\ tweet(u, \eta)]\!]_u$ *- Only u's followers can access her tweets.*
$P2(u) = [\![\neg S_{Ag \setminus followers(u) \setminus \{u\}} retweet(u, tu, \eta)]\!]_u$ *- Only u's followers can access her retweets.*
$P3(u) = [\![\neg S_{Ag \setminus \{u\}}\ location(u, \eta)]\!]_u$ *- No tweet by u contains her location.*
$P4(u) = \forall i.[\![\neg K_i\ (email(u) \vee phone(u)) \implies \neg P_i^u\ accessProfRec]\!]_u$ *- No user i can receive a recommendation to follow u, unless i knows u's email or phone number.*
$P5(u) = [\![\neg SP^u_{Advertisers}\ sendAd]\!]_u$ *- No advertisement companies can send ads to user u.*

In addition, users in Twitter are not allowed to have more than one instance of each type of privacy policy at the same time. Definition 12 formally describes the structure of the privacy policies accepted by the admissibility condition $\mathbb{RE}_{\text{Twitter}}$.

Auxiliary functions. The set of auxiliary functions consists of:

- $followers : Ag_{Twitter} \rightarrow 2^{Ag_{Twitter}}$ - This function returns the followers of a given user, i.e. for $u \in Ag_{\text{Twitter}}$, $followers(u) = \{\, i \mid (i, u) \in C_{Follower} \}$.
- $state : Ag_{Twitter} \rightarrow St$ - This function returns the state of a user's account, which can be *public* or *private*. For $u \in Ag_{\text{Twitter}}$, it returns *private* if the policies $P1, P2 \in \pi_u$ and *public* otherwise.
- $inclocation : Ag_{Twitter} \rightarrow Bool$ - This function returns the user's preference for revealing the location with the tweet. For $u \in Ag_{\text{Twitter}}$ it returns *false* if the policy $P3 \in \pi_u$, and *true* otherwise.
- $beingReco : Ag_{Twitter} \rightarrow Bool$ - This function returns the user's preference about being recommended to be followed by other users who have access her email or

phone number. For $u \in Ag_{\text{Twitter}}$ it returns *false* if the policy $P4 \in \pi_u$, and *true* otherwise.

- $getTweetInfo : Ag_{Twitter} \times \mathbb{N} \rightarrow 2^{\mathcal{P}_{Twitter}}$ - This function extracts information from a given tweet, for instance, the location ($location(o, \eta)$), the users mentioned in the tweet ($mention(u_1, o, \eta)$ $\ldots$ $mention(u_m, o, \eta)$), and the attached pictures ($picture(o, \eta, 1)$ $\ldots$ $picture(o, \eta, j)$), where $m, j \in \mathbb{N}$ are indexes. This information is returned as a set of predicates.
- $audience : \mathcal{P}_{Twitter} \rightarrow 2^{Ag_{Twitter}}$ - This function returns the audience of some piece of information, i.e. the agents who know that information. Given $p(\overrightarrow{t}) \in \mathcal{P}_{\text{Twitter}}$, $audience(p(\overrightarrow{t})) = \{ i \mid SN \vDash K_i p(\overrightarrow{t})\}$
- $info : Ag_{Twitter} \rightarrow 2^{\mathcal{P}_{Twitter}}$ - This function returns all the information of a given agent. Given an agent $u \in Ag_{\text{Twitter}}$, $info(u) = \{p(u, \overrightarrow{t}) | p(u, \overrightarrow{t}) \in KB_u\}$.

Properties of $\mathcal{FPPF}_{\textbf{Twitter}}$. The role of the properties is twofold. Firstly, they are used to encode some of the properties of the specific SNS and, secondly, some of these assumptions are added to the knowledge base *KB* of all the agents.

Note that, for the following set of properties, we write that an agent has access to a predicate $p(\overrightarrow{t})$ if she knows it, i.e., $K_i p(\overrightarrow{t})$. The intuition behind this choice is that if the agent "learnt" the predicate, it is because she had access to it. $\mathbb{A}_{\text{Twitter}}$ consists of the following properties:

- *Property 1.* If a user has access to a tweet, $tweet(o, \eta)$, then she can access all the information of that tweet. For all $p(\overrightarrow{t}) \in getTweetInfo(o, \eta)$,

$$\forall i.\forall o.\forall \eta.(K_i tweet(o, \eta) \implies K_i p(\overrightarrow{t}))$$

- *Property 2.* If a user has access to another user's tweet, $tweet(o, \eta)$, she can also access that user's profile.

$$\forall i.\forall o.\forall \eta.(K_i tweet(o, \eta) \implies P_i^o accessProf)$$

  This property models the fact that there is a link to the profile of the user who tweeted the tweet.

- *Property 3.* If a user has access to another user's retweet, $retweet(u, o, \eta)$, she can also access that user's profile and the owner of the tweet's profile.

$$\forall i.\forall u.\forall o.\forall \eta.(K_i retweet(u, o, \eta) \implies P_i^u accessProf \wedge P_i^o accessProf)$$

- *Property 4.* If a user has access to another user's favourite, $\mathit{favourite}(u, o, \eta)$, she can also access that user's profile and the owner of the tweet's profile.

$$\forall i.\forall u.\forall o.\forall \eta.(K_i \mathit{favourite}(u, o, \eta) \implies P_i^u \mathit{accessProf} \wedge P_i^o \mathit{accessProf})$$

Properties 2-4 may not seem very intuitive. They come from a design choice we make when implementing the behaviour of the SNS. In Twitter, when someone accesses another user's tweet, retweet or favourite, the user has the possibility of accessing the profiles of the owner of the tweet, retweet and favourite, respectively. The user only gets a chance to access the profile, because if that profile is not public only followers can access it, and this will be checked when the user is actually trying to access the profile. In our instantiation, we chose to model this using the permission operator and this is the reason why the mentioned properties give the permission to access the profile. A different approach could have been creating an attribute called $\mathit{profile}(u)$, which as soon as it is learnt by a user, it permits them to access u's profile. Moreover, the designer of the SNS can define as many properties as she considers necessary for the SNS, beyond the four properties introduced here.

III.3 Privacy Policies in Dynamic SNS

Social network users usually execute events. For example, they can post messages on a timeline, they can like a given post, share pictures, and so forth. Different events may change the knowledge, the permissions, or the connections between agents in the SNS. In this section, we formally incorporate the events that can be executed in the SNS and give the operational semantics rules for modelling the events' behaviour in $\mathcal{FPPF}$. These rules formally describe how SNMs change when a particular event occurs. This leads to having sets of SNMs, which represent the state of the SNS at a given moment in time. We also include a labelled transition system in $\mathcal{FPPF}$, which contains all the information about the evolution of the SNS. As in the previous section, we describe how these elements are instantiated for particular social networks and we conclude by extending the Twitter instantiation provided in Section III.2.

III.3.1 Labelled Transition System

Labelled Transition Systems (LTSs) have extensively been used in computer science to describe the behaviour of systems. In short, they are directed graphs where nodes represent states and edges the transitions between states. The edges are labelled with the name of the event which originates the transition between state.

In order to represent the behaviour induced by the events of the SNS, we define an LTS, and use it to keep track of the epistemic and deontic states as the SNS evolves. Nodes in the LTS represent *configurations*, which are SNMs. The set of all configurations in the LTS is a subset of all possible SNMs, $\mathcal{SN}$, since the LTS only contains the SNMs resulting from the execution of an event. Transitions in the LTS represent the evolution from a configuration to another, as a result of the execution of an event.

Definition 13. *An* SNS Labelled Transition System *(SNSLTS) is a tuple* $\langle Conf, EVT, \rightarrow, c_0\rangle$*, where*

- *$Conf$ is a set of social network models, $Conf \subseteq \mathcal{SN}$;*
- *EVT is the set of all possible events which can be executed in the SNS;*
- *$\rightarrow \subseteq Conf \times 2^{EVT} \times Conf$ is a transition relation;*
- *$c_0 \in Conf$ is the initial configuration of the social network.*

Given a set of events $E \subseteq EVT$ and the configurations $c_0, c_1 \in Conf$, we write $c_0 \xrightarrow{E} c_1$ to denote that the SNS evolves from c_0 to c_1 by the execution (in parallel) of the events in E. If E only contains one event, the transition represents a regular sequential execution. Note that the type of $\rightarrow$ allows for true parallelism in the execution of events. However we do not study possible side effects of the interleavings in the execution of parallel events, instead we will assume that the result of the execution in parallel is independent of the interleaving, leaving this issue as future work. For all configurations c it holds that $c \xrightarrow{\emptyset} c$.

Now we can formally define in $\mathcal{FPPF}$ dynamic SNSs as described by the Labelled Transition System.

Definition 14 (Dynamic $\mathcal{FPPF}$)**.** *The tuple* $\langle \mathcal{LTS}_{\mathcal{SN}}, \mathcal{KBL}, \models, \mathcal{PPL}, \models_C\rangle$ *is a* dynamic privacy policy framework *(denoted by $\mathcal{FPPF}^{\mathcal{D}}$), where*

- *$\mathcal{LTS}_{\mathcal{SN}}$ is the set of all possible SNS labelled transition systems;*
- *$\mathcal{KBL}$ is a knowledge-based logic;*
- *$\models$ is a satisfaction relation defined for $\mathcal{KBL}$;*
- *$\mathcal{PPL}$ is a formal language for writing privacy policies;*
- *$\models_C$ is a conformance relation defined for $\mathcal{PPL}$.* $\square$

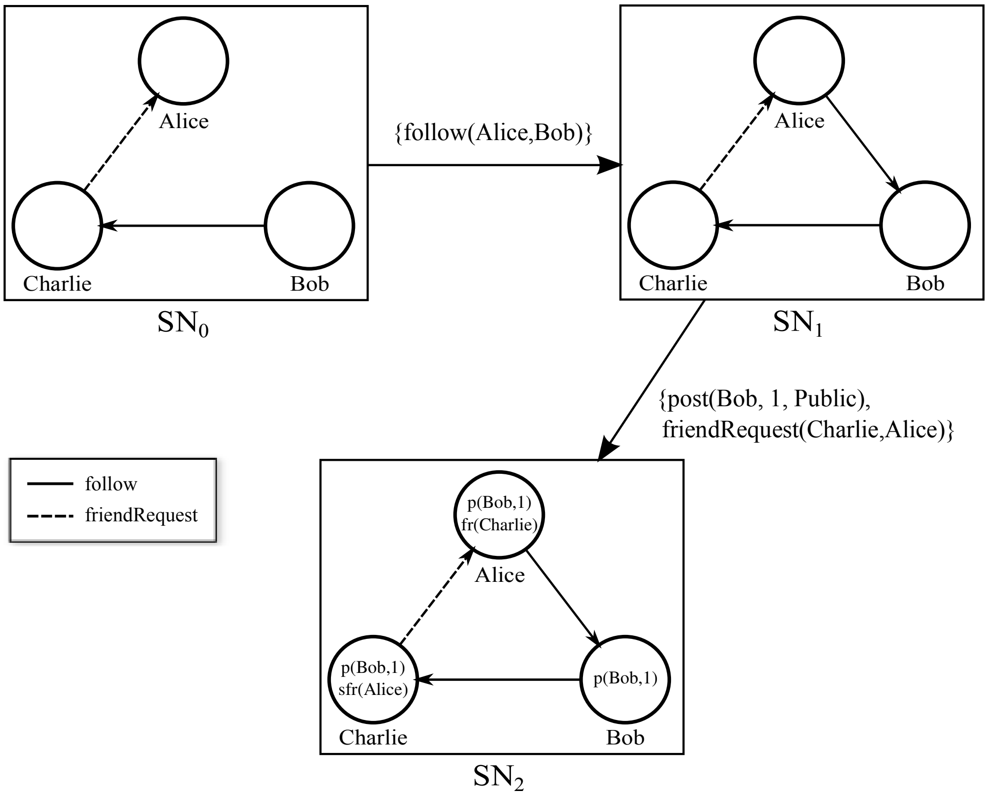


Figure III.3: Example of SNS Labelled Transition System

Example 7. *In Figure III.3 we give an example of an SNSLTS. This SNSLTS shows a possible sequence of events that can be executed. The rectangles represent 3 configurations $SN_0, SN_1, SN_2 \in Conf$. Each configuration depicts the SNM at different points in the execution. Since there are no events that involve the addition or removal of any users, all configurations have the same set of agents $Ag = \{Alice, Bob, Charlie\}$. SN_0 is the initial configuration. In this configuration, Bob follows Charlie, which is represented by a unidirectional arrow between them. The dashed arrow from Charlie to Alice expresses that Charlie is able to send a friend request to Alice.*

The transition from SN_0 to SN_1 represents that the SNS can evolve from the configuration SN_0 to SN_1 by executing of the event $follow(Alice, Bob)$, i.e. $SN_0 \xrightarrow{\{follow(Alice,Bob)\}} SN_1$. This event creates a new relation between Alice and Bob, which is modelled with a directed arrow between them in the resulting SNM, SN_1. This transition comprises *only one event, which means that no other event was executed in parallel. In SNSLTSs, transitions are labelled with sets of events representing the actions executed in parallel. In* $SN_1 \xrightarrow{\{post(Bob,1,Public), friendRequest(Charlie,Alice)\}} SN_2$ *two events are executed in parallel. On one hand, Bob posts his first post publicly in the SNS,* $post(Bob, 1, Public)$*. As a consequence, all users in* SN_2 *have learnt* $p(Bob, 1)$*, which is a predicate representing Bob's post. Formally,* $p(Bob, 1) \in KB_i$ *for all* $i \in Ag$*. In parallel, Charlie sends a friend request to Alice. Let us assume, for the sake of this example, that this is needed to check whether a friend request can be sent. Charlie is allowed to perform this event since*

$$SN_1 \vDash P^{Alice}_{Charlie} friendRequest$$

holds. Finally, the result of executing this event is that, in SN_2*, Alice knows that Charlie sent her the friend request,* $fr(Charlie) \in KB_{Alice}$*; and Charlie knows that he has sent the friend to Alice,* $sfr(Alice) \in KB_{Charlie}$*.*

Note that in the LTS of the previous example we can only observe the consequences of executing the events, but it is not possible to formally describe their behaviour. In the following sections we will introduce the dynamic instantiation of our framework and the operational semantics rules that formally define the behaviour of each event.

III.3.2 Dynamic $\mathcal{FPPF}_{\mathcal{S}}$

As in the static case, the dynamic instantiation of an SNS requires the specification of each of the components of the $\mathcal{FPPF}$ tuple. In particular, different SNSs will have different sets of events, EVT, which they can execute. Hence we can extend the definition of $\mathcal{FPPF}$ as follows:

Definition 15. *A* dynamic $\mathcal{FPPF}$ instantiation*, denoted as* $\mathcal{FPPF}^{\mathcal{D}}_{\mathcal{S}}$*, for a social network service* $\mathcal{S}$ *is an* $\mathcal{FPPF}_{\mathcal{S}}$*, in which the elements of the* $\mathcal{LTS}_{\mathcal{SN}}$ *are instantiated:*

$$\mathcal{FPPF}^{\mathcal{D}}_{\mathcal{S}} = \langle \mathcal{LTS}^{\mathcal{S}}_{\mathcal{SN}}, \mathcal{KBL}, \vDash, \mathcal{PPL}, \vDash_C \rangle$$

where $\mathcal{LTS}^{\mathcal{S}}_{\mathcal{SN}} = \langle Conf_{\mathcal{S}}, EVT_{\mathcal{S}}, \rightarrow_{\mathcal{S}}, c_0 \rangle$ *and*

- $Conf_{\mathcal{S}} \subseteq \mathcal{SN}_{\mathcal{S}}$ *is the set of all social network models for* $\mathcal{FPPF}^{\mathcal{D}}_{\mathcal{S}}$*;*
- $EVT_{\mathcal{S}}$ *is the set of all possible events in* $\mathcal{FPPF}^{\mathcal{D}}_{\mathcal{S}}$*;*
- $\rightarrow_{\mathcal{S}}$ *is the transition relation determined by the operational semantics rules for* $EVT_{\mathcal{S}}$*;*
- $c_0 \in Conf_{\mathcal{S}}$ *is the initial model of* $\mathcal{FPPF}^{\mathcal{D}}_{\mathcal{S}}$*.*

III.3.3 Operational Semantics Rules for SNS

The dynamic behaviour of an SNS is described in terms of a small step operational semantics. For every event in $EVT_{\mathcal{S}}$, there will be one or more operational semantics rules which describe its behaviour. The generic form of the rules is as follows:

$$\frac{Q_1 \dots Q_n}{SN \xrightarrow{e} SN'}$$

The premises $Q_1 \dots Q_n$ may be predicates, side conditions or any other auxiliary information used to describe the rule. They are defined by leveraging all the elements of $\mathcal{FPPF}$ and the instantiation $\mathcal{FPPF}_{\mathcal{S}}$ in which the rules are defined. The SNMs SN and SN' are tuples as defined in Definition 2, i.e., $\langle Ag, \mathcal{A}, KB, \pi \rangle$. The only elements of $\mathcal{A}$ involved in definition of operational semantics rules are connection predicates $\{C_i\}_{i \in \mathcal{C}}$, permission predicates $\{A_i\}_{i \in \Sigma}$ and the set of generic predicates $\mathcal{P}$. Therefore, we will write $\{\{C_i\}_{i \in \mathcal{C}}, \{A_i\}_{i \in \Sigma}, \mathcal{P}\}$ to refer to $\mathcal{A}$. For the sake of clarity, we will not explicitly write the rest of the elements of $\mathcal{A}$. Any element of the SNM tuple which is not involved in the execution of the rule will be replaced with "_". The operational semantics rules are divided in 4 types, as reported in Table III.3. We use the superindex e whenever an update of the SNM depends on event e. In what follows we first describe the intuition for each type of rule and then we provide a detailed description of an epistemic rule.

Epistemic. These rules are used to specify events that change the knowledge and/or the permissions of an SNM. As a result, the premises appearing in epistemic rules will only update the elements KB and A_i of the social network model involved in those rules. Agents' knowledge increases monotonically, for this reason, KB will only grow after the execution of an epistemic rule (see the first premise of the epistemic rule in Table III.3). Unlike knowledge, permissions can be granted or denied, which makes it possible for the pairs in the binary relations A_i to be added or removed.

Sometimes SNSs release information by making a message available to a group of users, e.g. tweets on Twitter or posts on Facebook. In dynamic epistemic logic this type of event is known as *public announcement* [7]. The result of performing a public announcement is that the disclosed information becomes *common knowledge* to the group of agents which are the audience of that announcement. We use common knowledge to accurately model what is known for everyone in a group.

Topological. These rules only affect the topology of SNMs. The social topology represents the elements of an SNM which come from the social graph. Therefore these

Epistemic

$$\frac{\begin{array}{c}\forall j \in Ag\ KB'_j = KB_j \cup \Gamma^e_j \text{ where } \Gamma^e_j \subseteq \mathcal{F}_{\mathcal{KBL}} \\ A'_i = (A_i \setminus \mathit{PerToRmv}^e) \cup \mathit{NewPer}^e \text{ where } \mathit{NewPer}^e \in 2^{Ag \times Ag} \text{ and } \mathit{PerToRmv}^e \in 2^{A_i} \\ P_1 \ldots P_m \in \mathcal{P} \text{ where } m \in \mathbb{N}\end{array}}{\langle \_, \{\_, \{A_i\}_{i\in\Sigma}, \mathcal{P}\}, KB, \_\rangle \xrightarrow{e} \langle \_, \{\_, \{A'_i\}_{i\in\Sigma}, \mathcal{P}\}, KB', \_\rangle}$$

Topological

$$\frac{\begin{array}{c}Ag' = (Ag \setminus \mathit{AgtToRmv}^e) \cup \mathit{NewAgt}^e \text{ where } \mathit{NewAgt}^e \in 2^{\mathcal{AU}} \text{ and } \mathit{AgtToRmv}^e \in 2^{Ag} \\ C'_i = (C_i \setminus \mathit{ConToRmv}^e) \cup \mathit{NewCon}^e \text{ where } \mathit{NewCon}^e \in 2^{Ag \times Ag} \text{ and } \mathit{ConToRmv}^e \in 2^{C_i} \\ P_1 \ldots P_m \in \mathcal{P} \text{ where } m \in \mathbb{N}\end{array}}{\langle Ag, \{\{C_i\}_{i\in\mathcal{C}}, \_, \mathcal{P}\}, \_, \_\rangle \xrightarrow{e} \langle Ag', \{\{C'_i\}_{i\in\mathcal{C}}, \_, \mathcal{P}\}, \_, \_\rangle}$$

Policy

$$\frac{\begin{array}{c}\forall j \in Ag\ \pi'_j = (\pi_j \setminus \mathit{PolToRmv}^e_j) \cup \mathit{NewPol}^e_j \text{ where } \mathit{NewPol}^e_j \in 2^{\pi_j} \text{ and } \mathit{PolToRmv}^e_j \subseteq \mathcal{F}_{\mathcal{PPL}} \\ P_1 \ldots P_m \in \mathcal{P} \text{ where } m \in \mathbb{N}\end{array}}{\langle Ag, \{\_, \_, \mathcal{P}\}, \_, \pi\rangle \xrightarrow{e} \langle Ag, \{\_, \_, \mathcal{P}\}, \_, \pi'\rangle}$$

Hybrid

$$\frac{\begin{array}{c}\forall j \in Ag\ KB'_j = KB_j \cup \Gamma^e_j \text{ where } \Gamma^e_j \subseteq \mathcal{F}_{\mathcal{KBL}} \\ A'_i = (A_i \setminus \mathit{PerToRmv}^e) \cup \mathit{NewPer}^e \text{ where } \mathit{NewPer}^e \in 2^{Ag \times Ag} \text{ and } \mathit{PerToRmv}^e \in 2^{A_i} \\ Ag' = (Ag \setminus \mathit{AgtToRmv}^e) \cup \mathit{NewAgt}^e \text{ where } \mathit{NewAgt}^e \in 2^{\mathcal{AU}} \text{ and } \mathit{AgtToRmv}^e \in 2^{Ag} \\ C'_i = (C_i \setminus \mathit{ConToRmv}^e) \cup \mathit{NewCon}^e \text{ where } \mathit{NewCon}^e \in 2^{Ag \times Ag} \text{ and } \mathit{ConToRmv}^e \in 2^{C_i} \\ \forall j \in Ag\ \pi'_j = (\pi_j \setminus \mathit{PolToRmv}^e_j) \cup \mathit{NewPol}^e_j \text{ where } \mathit{NewPol}^e_j \in 2^{\pi_j} \text{ and } \mathit{PolToRmv}^e_j \subseteq \mathcal{F}_{\mathcal{PPL}} \\ P_1 \ldots P_m \in \mathcal{P} \text{ where } m \in \mathbb{N}\end{array}}{\langle Ag, \{\{C_i\}_{i\in\mathcal{C}}, \{A_i\}_{i\in\Sigma}, \mathcal{P}\}, KB, \pi\rangle \xrightarrow{e} \langle Ag', \{\{C'_i\}_{i\in\mathcal{C}}, \{A'_i\}_{i\in\Sigma}, \mathcal{P}\}, KB', \pi'\rangle}$$

Table III.3: Generic structure of the operational semantics rules

rules update the sets Ag and C_i of an SNM. Using topological rules we can model the addition or removal of users and the relationships among them.

Policy. Policy rules allow to express changes in the privacy policies of the agents. Therefore the only element of the SNM that will be modified is π. As in the previous case, π may be updated by adding or removing privacy policies.

Hybrid. As the name suggests, these rules can be used in case an event in the SNS causes a mix of the previous types of rules to apply. Consequently, hybrid rules will combine premises of the three types and possibly update the SNM.

In order to clarify the specific meaning of the rules in Table III.3, here we provide a detailed explanation of the generic epistemic rule. The first premise in the rule,

$$\forall j \in Ag.\ (KB'_j = KB_j \cup \Gamma^e_j) \text{ where } \Gamma^e_j \subseteq \mathcal{F}_{\mathcal{KBL}}$$

represents the update of the knowledge bases of the agents. KB_j is the knowledge base of agent j before the execution of the event e and KB'_j is the resulting knowledge base after the execution of e. The new knowledge is given by Γ^e_j, which is defined to be a set of formulae $\mathcal{F}_{\mathcal{KBL}}$. Note that Γ is parametrised by the event e and the agent j, meaning that not all agents will have the same update of knowledge. Let $Ag = \{Alice, Bob\}$, then for an event e_1 that only updates Bob's knowledge with the predicate $p(\overrightarrow{t})$ we would have $\Gamma^{e_1}_{Alice} = \emptyset$ and $\Gamma^{e_1}_{Bob} = \{p(\overrightarrow{t})\}$.

The second premise in the generic epistemic rule expresses the update of permission in the SNM,

$$A'_i = (A_i \setminus PerToRmv^e) \cup NewPer^e$$

where $NewPer^e \in 2^{Ag \times Ag}$ and $PerToRmv^e \in 2^{A_i}$. A_i is the set of agents' pairs representing the set of permissions of type $i \in \Sigma$[2] before executing the event e, and after its execution A'_i will contain the resulting pairs. $PerToRmv^e$ represents the pairs to be removed in the SNM, its type is 2^{A_i} meaning that only existing pairs can be removed. $NewPer^e$ is the set with the new pairs of agents representing new permissions, since its type is $2^{Ag \times Ag}$ permission between any two agents can be created. When the event e only removes permission, then $NewPer^e = \emptyset$ (i.e. $A'_i = (A_i \setminus PerToRmv^e) \cup \emptyset$), on the other hand, if e only adds new permission $PerToRmv^e = \emptyset$. In general, any kind of update can be expressed. The same applies for updates of agents and connections in topological rules and policies in policy rules. The last premise

$$P_1 \dots P_m \in \mathcal{P} \text{ where } m \in \mathbb{N}$$

is used to express any other auxiliary predicate involving any element of $\mathcal{FPPF}$ required for the execution of the rule.

In what follows we show how to make use of the operational semantics rules to model the behaviour of a concrete SNS. Specifically, we will provide the set of rules for the events defined in a dynamic instantiation of Twitter.

[2]Remember that in SNMs we use binary relations between agents to represent permission (see Section III.2.1)

III.3.4 Dynamic Instantiation of Twitter

In this section we present the dynamic instantiation of Twitter, $\mathcal{FPPF}^{\mathcal{D}}_{\text{Twitter}}$, by extending the instantiation $\mathcal{FPPF}_{\text{Twitter}}$ introduced in Section III.2.4. The set EVT_{Twitter} contains all the relevant events for the privacy analysis of Twitter. Specifically, EVT_{Twitter} consists of the following elements:

- *tweet* - It is one the core events of Twitter. It is used to post some piece information.
- *retweet* - It is used to share an already tweeted tweet.
- *favourite* - It allows users to classify tweets as favourite.
- *accessProf* - It represents the action of accessing a user's profile.
- *createProf* - It is the first event a user executes for joining Twitter. The user is required to provide a set of basic information which determines her profile.
- *follow* - Users can connect with other users by means of the follower relationship.
- *acceptFollowReq* - When a user's profile is not public the *follow* event enables a request to the user. In order for the connection to be established the request must be accepted. This event represents the action of accepting the request.
- *block*, *unblock* - In Twitter a user can block other users and can revert this decision.
- *showReco* - Twitter shows a selection of recommended-to-follow users, when the email or the phone number of the recommended user is known by the one to whom the recommendation is shown.
- *showAdv* - This event models the action of a company sending an advertisement to a concrete user.
- *allowAdv*, *disallowAdv* - A user can (dis)allow a company from sending advertisement. These events model the activation and deactivation of this permission.
- *changeStPriv*, *changeStPub* - These events model the switching between *'Private'* or *'Public'* accounts.
- *inclLoc*, *notInclLoc* - These events represent whether the location is included or not in the tweet, respectively.

Tweet - T1

$$\frac{\begin{array}{c} Au = \mathit{followers}(u) \cup \{u\} \cup \{v \mid \mathit{mention}(v, u, \eta) \in \mathit{TweetInfo}\} \\ \mathit{state}(u) == \mathit{'Public'} \qquad \mathit{inclocation}(u) == \mathit{true} \\ \forall \varphi \in \mathit{TweetInfo}, \forall i \in Au\ \mathit{KB}_i' = \mathit{KB}_i \cup \{C_{Au}\varphi\} \end{array}}{\langle \_, \_, \mathit{KB}, \_ \rangle \xrightarrow{\mathit{tweet}(u, \mathit{TweetInfo})} \langle \_, \_, \mathit{KB}', \_ \rangle} \quad (\text{T1.1})$$

$$\frac{\begin{array}{c} Au = \mathit{followers}(u) \cup \{u\} \\ \mathit{state}(u) == \mathit{'Private'} \qquad \mathit{inclocation}(u) == \mathit{false} \qquad \mathit{location}(u, \eta) \notin \mathit{TweetInfo} \\ \forall \varphi \in \mathit{TweetInfo}, \forall i \in Au\ \mathit{KB}_i' = \mathit{KB}_i \cup \{C_{Au}\varphi\} \end{array}}{\langle \_, \_, \mathit{KB}, \_ \rangle \xrightarrow{\mathit{tweet}(u, \mathit{TweetInfo})} \langle \_, \_, \mathit{KB}', \_ \rangle} \quad (\text{T1.2})$$

$$\frac{\begin{array}{c} Au = \mathit{followers}(u) \cup \{u\} \cup \{v \mid \mathit{mention}(v, tu, \eta) \in \mathit{TweetInfo}\} \\ \mathit{state}(u) == \mathit{'Public'} \qquad \mathit{inclocation}(u) == \mathit{false} \qquad \mathit{location}(u, \eta) \notin \mathit{TweetInfo} \\ \forall \varphi \in \mathit{TweetInfo}, \forall i \in Au\ \mathit{KB}_i' = \mathit{KB}_i \cup \{C_{Au}\varphi\} \end{array}}{\langle \_, \_, \mathit{KB}, \_ \rangle \xrightarrow{\mathit{tweet}(u, \mathit{TweetInfo})} \langle \_, \_, \mathit{KB}', \_ \rangle} \quad (\text{T1.3})$$

$$\frac{\begin{array}{c} Au = \mathit{followers}(u) \cup \{u\} \qquad \mathit{state}(u) == \mathit{'Private'} \qquad \mathit{inclocation}(u) == \mathit{true} \\ \forall \varphi \in \mathit{TweetInfo}, \forall i \in Au\ \mathit{KB}_i' = \mathit{KB}_i \cup \{C_{Au}\varphi\} \end{array}}{\langle \_, \_, \mathit{KB}, \_ \rangle \xrightarrow{\mathit{tweet}(u, \mathit{TweetInfo})} \langle \_, \_, \mathit{KB}', \_ \rangle} \quad (\text{T1.4})$$

Create Profile - T2

$$\frac{\begin{array}{c} u \notin Ag \qquad Ag' = Ag \cup \{u\} \qquad \mathit{KB}_i' = \mathit{InitialInfo} \\ \forall j \in Advertisers\ A'_{sendAd} = A_{sendAd} \cup \{(u, j)\} \qquad A'_{accessProf} = A_{accessProf} \cup \{(u, u)\} \end{array}}{\langle Ag, \{\_, \{A_i\}_{i \in \Sigma}, \_\}, \mathit{KB}, \_ \rangle \xrightarrow{\mathit{createProf}(u, \mathit{InitialInfo})} \langle Ag', \{\_, \{A_i'\}_{i \in \Sigma}, \_\}, \mathit{KB}', \_ \rangle} \quad (\text{T2.1})$$

Table III.4: Create and Tweet rules for $\mathcal{FPPF}^{\mathcal{D}}_{\text{Twitter}}$

We instantiate the rules in Table III.3 for each of the events described above. As a result, we obtain the operational semantics rules for the social network. In what follows we describe the Twitter rules for the events *createProf* and *tweet* detailed in Table III.4. For the full set of rules modelling Twitter semantics please refer to Appendix A.3-A.6. Note that in the previous rules we use "=" for assignments and "==" for equality.

The event *createProf* describes how the social network model changes when a new user joins the SNS, i.e., $SN \xrightarrow{\mathit{createProf}(u,\mathit{InitialInfo})} SN'$ for $SN, SN' \in \mathcal{SN}_{\text{Twitter}}$, $u \in Ag$ and $\mathit{InitialInfo} \subseteq \mathcal{F}_{\mathcal{KBL}}$ (representing the initial set of information that users provide in Twitter). Rule (T2.1) consists of one condition, which if satisfied, leads to four consequences. The condition $u \notin Ag$ requires that the new user is not already registered, i.e., her node does not exist in the SNM before executing the event. The remaining premises represent the effects of executing the event. Firstly, $Ag' = Ag \cup \{u\}$ (where $Ag' \in SN'$), specifies that the new user is added to the SNM. Secondly, $KB'_i = \mathit{InitialInfo}$, represents that in the new SNM SN', the user knows all the information she provided when signing up. Moreover the user is able to access her own profile as represented by $A'_{accessProf} = A_{accessProf} \cup \{(u,u)\}$. Finally, $\forall j \in Advertisers\ A'_{sendAd} = A_{sendAd} \cup \{(u,j)\}$, models the set of advertisers, $Advertisers \subseteq Ag$, who can send advertisements to the user.

In general, an event may give rise to more than one operational semantics rule. *tweet* is an example of such an event (see Table III.4). It is composed by 4 rules, which determine its behaviour depending on certain conditions. These conditions consider whether a user has protected her tweets and whether she allows her location to be included in her tweets. Since the policies can be either activated or deactivated, this leads to four different social network models after its execution. Suppose that $SN \xrightarrow{tweet(u,\, \mathit{TweetInfo})} SN'$ for $SN, SN' \in \mathcal{SN}_{\text{Twitter}}$, $u \in Ag$ and $\mathit{TweetInfo} \in 2^{\mathcal{P}_{\text{Twitter}}}$ (representing the information disclosed in the tweet, i.e., location of the tweet, mentions, pictures, etc). In the first line of all the rules for *tweet* we specify what will be the audience of the tweet. This depends on the type of the account of the user who is tweeting. If the state of the user's account is *'Public'*, then the tweet will be disclosed to her followers and to the people mentioned in the tweet, $\mathit{followers}(u) \cup \{u\} \cup \{v | \mathit{mention}(v, u, \eta) \in \mathit{TweetInfo}\}$ (rules (T1.1) and (T1.3)). Otherwise, the audience is restricted to only her followers $\mathit{followers}(u) \cup \{u\}$ (rules (T1.2) and (T1.4)). If the tweet location is deactivated, $\mathit{inclocation}(u) == \mathit{false}$, then the rules contain one extra condition which explicitly requires that the location should not be part of the information disclosed in the tweet, $\mathit{location}(u, \eta) \notin \mathit{TweetInfo}$ (rules (T1.2) and (T1.3)). As a result, all the formulae describing the tweet information become common knowledge among the agents of the audience, $\forall \varphi \in \mathit{TweetInfo}, \forall i \in Au\ K'_i = K_i \cup \{C_{Au}\varphi\}$.

The reader may wonder why the audience of a tweet is not all Twitter users when the profile of the tweet's owner is public. The reason is because we want to model the exact behaviour of the SNS. In Twitter when a user (with a public profile) tweets a message, this message is shown in her followers' timeline. Additionally, since the profile is public, any other user (who is not following her) can check all her tweets. This is modelled with the event *accessProf*. The rule modelling the event's behaviour consists of 2 cases, which distinguish if the user has a public or a private profile. If the profile is public any user which executes the events will get access to all the tweets. For the formal definition of this rule see III.A.1.

III.4 Proving Privacy in Social Networks

The dynamic part of $\mathcal{FPPF}$ raises new questions about the privacy of the SNS. The execution of an event can lead to a state of the social network in which some privacy policies are violated. As a designer, one may want to be sure that all the events implemented in the SNS preserve the set of privacy policies that users have defined. In this section, we define the notion of privacy-preserving SNS, which, in short, expresses that all privacy policies must be in conformance with the SNS at any point in the execution. This concept allows us to formally analyse the privacy of SNSs modelled in $\mathcal{FPPF}$. As an example, we describe how to carry out a privacy analysis of Twitter and Facebook.

III.4.1 Does an SNS preserve privacy?

In SNSs privacy policies can be violated because of the execution of many events. Therefore, in order to make sure that all privacy policies will be preserved in the SNS, we have to ensure that none of the events can violate any of the privacy policies. Since in $\mathcal{FPPF}^{\mathcal{D}}$ we model the evolution of the SNS, we can formally prove whether the execution of the events defined in an SNS will preserve a set of privacy policies. We formalise this privacy condition as follows.

Definition 16. *An SNS $\mathcal{S}$ is* privacy-preserving *iff given a dynamic instantiation $\mathcal{FPPF}^{\mathcal{D}}_{\mathcal{S}}$ of $\mathcal{S}$, for any $SN, SN' \in \mathcal{SN}_{\mathcal{S}}$, $e \in EVT_{\mathcal{S}}$ and $\pi' \in \Pi_{\mathcal{S}}$ the following holds:*

$$\textit{If } SN \vDash_C \pi' \textit{ and } SN \xrightarrow{e} SN' \textit{ then } SN' \vDash_C \pi'$$

In the following sections we show whether this property holds for different sets of privacy policies in Twitter and Facebook.

III.4.2 Privacy in Twitter

Using the dynamic instantiation of Twitter that we defined in the previous section, $\mathcal{FPPF}^{\mathcal{D}}_{\text{Twitter}}$, we show that the described events in EVT_{Twitter} and the proposed specification using the operational semantics rules are privacy-preserving (as defined in Definition 16) with respect to the set of privacy policies of Twitter.

Theorem 1. *Twitter is privacy-preserving.*

Proof sketch: We check that the execution of none of the events in EVT_{Twitter} can violate any of the privacy policies in Π_{Twitter} by considering all possible combinations

of events and privacy policies (i.e. ensuring that Definition 16 holds). Here we only show the case when *tweet* (see Table III.4) is executed and the privacy policy $P1(u) = [\![\neg S_{Ag\setminus followers(u)\setminus\{u\}}\ tweet(u,\eta)]\!]_u$ is activated. We follow the same strategy for the remaining cases (see III.A.2 for the full detailed proof).
1. Given
1.1. $u \in Ag$ (owner of the privacy policy $P1(u)$)
1.2. Predicates to be disclosed $TweetInfo \subseteq 2^{\mathcal{P}}$ where $tweet(u,\eta) \in TweetInfo$
1.3. $e = tweet(u, TweetInfo)$
1.4. We want to prove:

$$\text{If } SN \vDash_C P1(u) \text{ and } SN \xrightarrow{e} SN' \text{ then } SN' \vDash_C P1(u)$$

2. By contradiction, let us assume
2.1. $SN \vDash_C P1(u)$ and $SN \xrightarrow{e} SN'$
2.2. $SN' \nvDash_C P1(u)$

3. By 2.2.
3.1. $SN' \nvDash_C P1(u)$ [Definition $\vDash_C$]
3.2. $SN' \vDash \neg\neg S_{Ag\setminus followers(u)\setminus\{u\}} tweet(u,\eta)$ $[\neg\neg_e]$
3.3. $SN' \vDash S_{Ag\setminus followers(u)\setminus\{u\}} tweet(u,\eta)$

4. By 3.3. and the definition of $\vDash$ we have
4.1. $\exists i \in Ag \setminus followers(u) \setminus \{u\}$ s.t. $SN' \vDash K_i tweet(u,\eta)$

5. By Definition of *tweet*, we have that
5.1. $\forall p(\overrightarrow{t}) \in TweetInfo\ SN' \vDash C_{followers(u)\cup\{u\}} p(\overrightarrow{t})$ [By 1.2.]
5.2. $SN' \vDash C_{followers(u)\cup\{u\}} tweet(u,\eta)$ [By $\vDash$]
5.3. $SN' \vDash E^0_{followers(u)\cup\{u\}} tweet(u,\eta) \wedge$
$E^1_{followers(u)\cup\{u\}} tweet(u,\eta) \wedge$
$E^2_{followers(u)\cup\{u\}} tweet(u,\eta) \wedge$
$E^3_{followers(u)\cup\{u\}} tweet(u,\eta) \wedge \ldots$ [By $\vDash$]
5.4. $SN' \vDash E^1_{followers(u)\cup\{u\}} tweet(u,\eta)$ [By $\vDash$]
5.5. $\forall j \in followers(u) \cup \{u\}\ SN' \vDash K_j tweet(u,\eta)$

6. By 2.1. we have
6.1. $SN \vDash_C P1(u)$ [By $\vDash_C$]
6.2. $SN \vDash \neg S_{Ag\setminus followers(u)\setminus\{u\}} tweet(u,\eta)$ [By Definition S_G]
6.3. $SN \vDash \neg(\bigvee_{i\in Ag\setminus followers(u)\setminus\{u\}} K_i tweet(u,\eta))$ [Morgan]

6.4. $SN \vDash \bigwedge_{i \in Ag \setminus followers(u) \setminus \{u\}} \neg K_i tweet(u, \eta)$

7. By 6.4. and 5.5. we have
7.1. $SN' \vDash_C P1(u)$

8. By 2.2. and 7.1. we derive a contradiction. □

The proof of Theorem 1 is carried out over the instantiation we constructed from the observable behaviour of Twitter. Having access to the source code would make it possible to define a more accurate instantiation of Twitter. Nevertheless it formally guarantees that an implementation which precisely behaves as described by the operational semantics rules will preserve all privacy policies defined for Twitter.

As we mentioned in Section VII.1, developers add new functionalities every day. Sometimes new privacy policies are added as well. Making sure that all privacy policies are effectively enforced in such a dynamic environment is a very difficult task.

Suppose Twitter developers decide to offer the following new privacy policy to their users: *"It is not permitted that I am mentioned in a tweet which contains a location"*. This privacy policy can be expressed in $\mathcal{PPL}$ as follows:

$$P6(u) = \forall i.\forall o.\forall \eta.[\![\neg(K_i location(o, \eta) \wedge K_i mention(u, o, \eta))]\!]_u.$$

Here we use $\mathcal{FPPF}^{\mathcal{D}}_{\text{Twitter}}$ to formally show that this privacy policy would not be enforced under the current operational semantics.

Lemma 1. *Twitter is not privacy-preserving if* $P6(u) \in \mathbb{RE}_{Twitter}$ *where* $u \in Ag_{Twitter}$.

Proof Sketch: Assume a user $u \in Ag$ who has never been mentioned and has one instance of $P6(u)$ in her set of policies, and another user $o \in Ag$ who executes the event

$$e = tweet(o, \{tweet(o, \eta), mention(u, o, \eta), location(o, \eta)\}).$$

Let us assume that $SN \xrightarrow{e} SN'$. From the assumptions we know that $SN \vDash_C P6(u)$, but according to the operational semantics of *tweet*, all users in the audience of the tweet will learn $mention(u, o, \eta)$ and $location(o, \eta)$ and therefore $SN' \nvDash_C P6(u)$. See III.A.2 for the detailed proof. □

Lemma 1 is an expected result. Twitter was not developed with $P6$ in mind. Yet the proof directly points to the event violating it. It also provides useful information of how the behaviour of Twitter should be modified to support $P6$.

III.4.3 What about Facebook?

Together with Twitter, Facebook is one of the giants of social media. Facebook connects millions of users who share information through events similar to the ones presented for Twitter. In this section, we use Facebook as target SNS to show yet another example of how $\mathcal{FPPF}$ can be used to analyse the privacy implications of adding new privacy policies.

In Facebook, when someone tags a user in a picture only the owner of the picture is required to confirm the tag. No confirmation from the tagged user is required. The only control the tagged user has over the tag is to hide the picture from her timeline or remove it after the tagging has been carried out. We model this behaviour in a reduced instantiation of Facebook, denoted as $\mathcal{FPPF}_{\text{FB-Tag}}$, which exclusively contains the required elements to model the tagging process.

Given $o, tge, tgr \in Ag_{\text{FB-Tag}}$ and $\eta \in \mathbb{N}$, the set of predicates, $\mathcal{P}_{\text{FB-Tag}}$, is composed by:

- $picture(o, \eta)$ - Picture η published by user o.
- $tagRequest(tgr, tge, o, \eta)$ - Tag request from the tagger (tgr) of the tagged user, taggee (tge), in picture $picture(o, \eta)$.
- $tag(tge, tgr, o, \eta)$ - Tag created by the tagger (tgr) of the tagged user, taggee (tge), in picture $picture(o, \eta)$.

The connections set only contains the friendship relationship, $\mathcal{C}_{\text{FB-Tag}} = \{C_{Friendship}\}$. The action $removeTag_{tag(tge,tgr,accepter,\eta)}$ is the only one included in the set $A_{\text{FB-Tag}}$. This action defines which users have permission to remove the tag $tag(tge, tgr, accepter, \eta)$. Regarding the auxiliary functions we only include:

- $audience : \mathcal{P}_{\text{FB-Tag}} \to 2^{Ag_{\text{FB-Tag}}}$ - As in Twitter, the *audience* function returns the audience of some piece of information. Given $p(\vec{t}) \in \mathcal{P}_{\text{FB-Tag}}$, $audience(p(\vec{t})) = \{\, i \mid SN \vDash K_i p(\vec{t})\}$.
- $friends : Ag_{\text{FB-Tag}} \to 2^{Ag_{\text{FB-Tag}}}$ - This function returns all the friends of a given user. Given $u \in Ag_{\text{FB-Tag}}$, $friends(u) = \{i | (u, i) \in C_{Friendship}\}$.

The previous elements constitute the static part of our (reduced) instantiation of Facebook, $\mathcal{FPPF}_{\text{FB-Tag}}$. In order to model the behaviour of the tagging event, we extend $\mathcal{FPPF}_{\text{FB-Tag}}$ with the operational semantics rules for the events *tag* and *acceptTagRequest*—both included in $EVT_{\text{FB-Tag}}$—as specified in Table III.5 thus completing the definition of $\mathcal{FPPF}^{\mathcal{D}}_{\text{FB-Tag}}$. The intuition behind the operational semantics rules is as follows.

Tag - FR1

$$\frac{picture(o,\eta) \in KB(tgr) \qquad KB'(o) = KB(o) \cup \{C_{\{o,tgr\}}\,tagRequest(tgr, tge, o, \eta)\} \\ KB'(tgr) = KB(tgr) \cup \{C_{\{o,tgr\}}\,tagRequest(tgr, tge, o, \eta)\}}{SN \xrightarrow{tag(tgr,tge,picture(o,\eta))} SN'} \quad \text{(FR1.1)}$$

Accept tag request - FR2

$$\frac{Au = audience(picture(o,\eta)) \cup friends(tge) \\ a = removeTag_{tag(tge,tgr,o,\eta)} \qquad acptr == o \\ tagRequest(tge, tgr, o, \eta) \in KB(acptr) \\ A'_a = A_a \cup \{(o,o),(o,tge)\} \qquad \forall i \in Au\; KB'(i) = KB(i) \cup \{C_{Au} tag(tge, tgr, o, \eta)\}}{SN \xrightarrow{acceptTagRequest(acptr,tge,tgr,picture(o,\eta))} SN'} \quad \text{(FR2.1)}$$

Table III.5: Tagging - Operational Semantics of Facebook

The event $tag(tgr, tge, picture(o,\eta))$ represents what happens when a (*tagger*), *tgr*, tags another user (*taggee*), *tge*, in a picture $picture(o,\eta)$. The tagger *tgr* must have access to the picture. We represent this by imposing the condition $picture(o,\eta) \in KB(tgr)$ in FR1.1. If the condition is satisfied, a tag request, informing that *tgr* wants to tag *tge* in $picture(o,\eta)$, is sent to the owner of the picture and it becomes common knowledge for both of them, so $\forall i \in \{o, tgr\}$ we have that

$$KB'(i) = KB(i) \cup \{C_{\{o,tgr\}}\,tagRequest(tgr, tge, o, \eta)\}$$

Note that the approval from the tagged user is not required.

The event $acceptTagRequest(acptr\ tge,\ tgr,\ picture(o,\ \eta))$ describes the result of accepting a tag request. The tag request must have been sent beforehand. The owner of the picture is the only user able to accept the tag, i.e., $acptr == o$, therefore it is required to check that the user accepting the tag has access to the tag request,

$$tagRequest(tge, tgr, o, \eta) \in KB(acptr).$$

The owner of the picture and the taggee will be permitted to remove the tag, which is specified as follows, given $a = removeTag_{tag(tge,tgr,o,\eta)}$

$$A'_a = A_a \cup \{(o,o),(o,tge)\}.$$

Also the tag is disclosed to the users part of the audience of the picture, thus becoming common knowledge among them. $\forall i \in \mathit{audience}(\mathit{picture}(o, \eta))$

$$KB'(i) = KB(i) \cup \{C_{Au} \mathit{tag}(\mathit{tge}, \mathit{tgr}, o, \eta)\}.$$

Suppose now that Facebook developers decide to offer to their users a better control over their tags by adding the following privacy policy:

"I can only be tagged in a picture if I have approved it".

We denote this privacy policy as $\mathit{FP1}(u)$ where $u \in Ag_{\text{FB-Tag}}$ and it is expressed in $\mathcal{PPL}$ as follows:

$$\forall o.\forall t.\forall \eta.[\![\neg K_u \mathit{tagRequest}(t, u, o, \eta) \implies \neg S_{Ag} \mathit{tag}(u, t, o, \eta)]\!]_u$$

meaning that for all pictures posted by a user o ($\mathit{picture}(o, \eta)$ where $\eta \in \mathbb{N}$), if the user u (the one who is going to be tagged) did not receive the tag request, then the tagging will not be carried out. By forcing u to be the one receiving the tag request, we ensure that it is u the one approving the tag.

As in Twitter, the following holds:

Lemma 2. *Facebook is not privacy-preserving if $\mathit{FP1}(u) \in \mathbb{RE}_{\mathit{FB\text{-}Tag}}$ where $u \in Ag_{\mathit{FB\text{-}Tag}}$.*

Proof sketch: Let $\mathit{tge} \in Ag$ be a user who has never been tagged and let $\mathit{tgr} \in Ag$ be a user who has executed the event $\mathit{tag}(\mathit{tgr}, \mathit{tge}, o, \eta)$ in order to tag tge in $\mathit{picture}(o, \eta)$ where $o \in Ag$ and $\eta \in \mathbb{N}$. The owner of $\mathit{picture}(o, \eta)$ is o. Assume a social network model, SN, where it holds that $\mathit{tagRequest}(\mathit{tge}, \mathit{tgr}, o, \eta) \in KB(o)$. In order for $\mathcal{FPPF}^{\mathcal{D}}_{\text{FB-Tag}}$ to preserve privacy it must hold that if $SN \xrightarrow{acceptTagRequest(o, tge, tgr, picture(o, \eta))} SN'$ and $SN \vDash_C \mathit{FP1}(\mathit{tge})$, then $SN' \vDash_C \mathit{FP1}(\mathit{tge})$ where $SN, SN' \in \mathcal{SN}_{\text{FB-Tag}}$.

Since tge was not tagged before the execution of $\mathit{FR2}$ we know that $SN \vDash_C \mathit{FP1}(\mathit{tge})$. Also since $\mathit{tagRequest}(\mathit{tge}, \mathit{tgr}, o, \eta) \in KB(o)$ and $\mathit{acptr} == o$ we know that $\mathit{FR2}$ can be executed. By the definition of $\mathit{FR2}$, we know that $SN' \vDash E_{Au} \mathit{tag}(\mathit{tge}, o, o, \eta)$, hence $SN' \nvDash_C \mathit{FP1}(\mathit{tge})$, which contradicts our claim $SN' \vDash_C \mathit{FP1}(\mathit{tge})$ and therefore the instantiation $\mathcal{FPPF}^{\mathcal{D}}_{\text{FB-Tag}}$ is not privacy-preserving. See III.A.2 for the detailed proof.

□

In short, the proof shows that the policy is not enforced because the owner of the picture can accept tags (FR2.1) of any user without their approval in any of her pictures. In this instantiation, since there are only two operational semantics rules, it is easy to discuss a possible modification in the rules so that *FP1* is supported.

Tag - FR1

$$\frac{\begin{array}{c}\boldsymbol{FP1(tge) \notin \pi_{tge}}\\ picture(o,\eta) \in KB(tgr) \qquad KB'(o) = KB(o) \cup \{C_{\{o,tgr\}} tagRequest(tgr, tge, o, \eta)\}\\ KB'(tgr) = KB(tgr) \cup \{C_{\{o,tgr\}} tagRequest(tgr, tge, o, \eta)\}\end{array}}{SN \xrightarrow{tag(tgr,tge,picture(o,\eta))} SN'} \tag{FR1.1}$$

$$\frac{\begin{array}{c}\boldsymbol{FP1(tge) \in \pi_{tge}} picture(o,\eta) \in KB(tgr)\\ \boldsymbol{KB'(tge) = KB(tge) \cup \{C_{\{tge,tgr\}} tagRequest(tgr, tge, o, \eta)\}}\\ KB'(tgr) = KB(tgr) \cup \{C_{\{tge,tgr\}} tagRequest(tgr, tge, o, \eta)\}\end{array}}{SN \xrightarrow{tag(tgr,tge,picture(o,\eta))} SN'} \tag{FR1.2}$$

Accept tag request - FR2

$$\frac{\begin{array}{c}Au = audience(picture(o,\eta)) \cup friends(tge) \qquad \boldsymbol{FP1(tge) \notin \pi_{tge}}\\ a = removeTag_{tag(tge,tgr,o,\eta)} \qquad acptr == o \qquad tagRequest(tge, tgr, o, \eta) \in KB(acptr)\\ A'_a = A_a \cup \{(o,o),(o,tge)\} \qquad \forall i \in Au\ KB'(i) = KB(i) \cup \{C_{Au} tag(tge, tgr, o, \eta)\}\end{array}}{SN \xrightarrow{acceptTagRequest(acptr,tge,tgr,picture(o,\eta))} SN'} \tag{FR2.1}$$

$$\frac{\begin{array}{c}Au = audience(picture(o,\eta)) \cup friends(tge)\\ \boldsymbol{FP1(tge) \in \pi_{tge}} \qquad a = removeTag_{tag(tge,tgr,o,\eta)}\\ \boldsymbol{acptr == tge} \qquad tagRequest(tge, tgr, o, \eta) \in KB(acptr)\\ A'_a = A_a \cup \{(o,o),(o,tge)\} \qquad \forall i \in Au\ KB'(i) = KB(i) \cup \{C_{Au} tag(tge, tgr, o, \eta)\}\end{array}}{SN \xrightarrow{acceptTagRequest(acptr,tge,tgr,picture(o,\eta))} SN'} \tag{FR2.2}$$

Table III.6: New Tagging rules supporting FP1

First of all, FR2.1 must guarantee that the taggee is accepting the tag if the policy is activated. In order to preserve this condition, we would need to replace $acptr == o$ with $acptr == tge$, which forces the taggee to be the one accepting the tag. Finally, FR1.1 must be slightly modified, since now the tag request will be sent to the taggee instead of the owner of the picture. Therefore,

$$KB'(o) = KB(o) \cup \{C_{\{o,tgr\}} tagRequest(tgr, tge, o, \eta)\}$$

is replaced with

$$KB'(tge) = KB(tge) \cup \{C_{\{tge,tgr\}} tagRequest(tgr, tge, o, \eta)\}.$$

The resulting operational semantics rules are presented in table III.6.

Finally, including the new two rules in $\mathcal{FPPF}^{\mathcal{D}}_{\text{FB-Tag}}$ and assuming that the only privacy policy in the instantiation is *FP1*, the following lemma holds:

Lemma 3. *Facebook is privacy-preserving.*

Proof sketch: We consider all possible rules that can be executed and show that none of them will violate *FP1*, which is the only policy available in the instantiation $\mathcal{FPPF}^{\mathcal{D}}_{\text{FB-Tag}}$. The only rule that can violate *FP1* is *FR2* (specifically the case *FR2.2*). Due to the similarity to the proof for Theorem 1 we omit the details here, but we follow the same strategy, i.e. we show by contradiction that *FP1* cannot be violated. We refer the reader to III.A.2 for the complete proof. □

III.5 Discussion and Related Work

In this section we describe applications of epistemic logic in security and other approaches to modelling SNSs. We also discuss the formalism developed by Fong *et al.* which describes the access control mechanisms present in most SNSs nowadays. Finally, we discuss the relation between $\mathcal{FPPF}$ and epistemic logic.

III.5.1 Epistemic Logic and SNSs

In the past, epistemic logic has been widely used for analysing security and privacy properties in multi-agent systems (MAS). Traditionally the evolution of knowledge in epistemic logic is modelled by means of runs and events, in the "run-and-systems" framework, known as *Interpreted Systems* [29].

Halpern *et al.* [38] use Interpreted Systems to formalise the notion of secrecy in MAS. They redefine the possibilistic and probabilistic security properties in epistemic logic, in the form of a modal operator which allows them to reason about knowledge, nondeterminism and probability together. Interpreted Systems also appear in [5], where Balliu presents a knowledge-based account to specify information flow conditions in a distributed setting. The main advantage of this approach is that it is able to express complex policies based on epistemic logic. One of the main drawbacks of Interpreted Systems is the high complexity of the model-checking. Nevertheless it has been studied how to implement efficient model-checkers which make it possible to verify properties of real systems. For instance, MCK [33] and MCMAS [53] are state of the art model checkers for temporal-epistemic logics based on Interpreted systems. They have successfully been used to verify security properties for several cryptographic protocols. We are not aware of any specific use for verifying privacy policies.

Interpreted systems allow to represent the knowledge at different points in time. There is no formal definition of the events that can be executed in order to specify how knowledge evolves. Instead they require a description of the protocol which models the evolution of knowledge. *Dynamic Epistemic Logic* (DEL) provides a basis for operations on knowledge evolution in epistemic logic [7, 74]. DEL encodes informational events by defining update operations over the classical Kripke models in epistemic logic. The most important feature with respect to the work carried out for this paper is the *public announcement*, which consists in the action of disclosing a piece of information to a set of agents.

It has recently been studied how to model the propagation of knowledge over the agents of an SNS by using DEL. In [84] Seligman *et al.* define *dynamic epistemic friendship logic* (DEFL), which on one hand, extends the classical Kripke model for epistemic logic with the information about the friendship relationships, and on the other hand, uses DEL to encode public and private announcements in the SNS. A private announcement is a disclosure of information between two agents, in which only the two involved agents are aware of the fact that the announcement occurred. In [80], DEL has been used to study, by means of a formal technique, the effect "Revolt or Stay-at-Home" in SNSs. This effect represents how the fact of knowing how many people (or agents) are going to revolt could influence our own decision to revolt or stay at home.

DEL turns out to be not well-suited to our setting. Firstly, because it is based in the classical Kripke semantics for epistemic logic [29]. As we describe in Section III.5.3, there are properties of knowledge that need to be further studied before we can encode SNMs in Kripke structures. Secondly, DEL is only used for modelling the evolution of knowledge, in our framework apart from epistemic rules, we allow for topological, policy and hybrid rules. Finally, and most importantly, the events defined in DEL are not equipped with conditions, i.e. the execution of events does not depend on the knowledge of the agents. By contrast, the execution of events in SNSs depends not only on the agents' knowledge, but also other network-dependent factors. As described in Section III.3.3, we use the premises of the rules when stating the conditions for each event.

III.5.2 Relationship-based Access Control

Currently SNS users share their resources by using the so called Relationship-Based Access Control (ReBAC). This paradigm gives access to user resources depending on her relationships with the owner of the resource. Fong *et al.* introduce a formalism that aims at providing a better understating of ReBAC [32, 31]. They develop a general formalism which can be instantiated, first in mono-relational social networks, e.g.

Facebook-like networks where the relationship between agents is friendship, and later in a more general setting, with poly-relational social networks where the type of the relationship is also taken into account (e.g. patient-physician, parent-child). In addition, they introduce the notion of *access contexts*, defined as a hierarchy of contexts to enable an inheritance mechanism of relationships. Hence the access to the resources also offers the possibility of articulating relationships between users depending on the access context. The audience of the resources is defined by means of ReBAC policies. In [12] Bruns *et al.* provide a language based on a hybrid logic which extends Fong's work and supports interesting policy idioms.

By contrast to our work, the ReBAC paradigm is not able to detect appropriately implicit disclosure of information. For example, if a user posts the location of another user, the latter has no control over the audience of her location. Therefore, the owner of the post defines the audience of another user's location. In our framework, the structure of the predicates can encode the actual owner of a resource independently of the user disclosing the information. Due to that, a user can later define a privacy policy which would protect a particular piece of her information, independently of who was the user disclosing that information. We claim that $\mathcal{FPPF}$ is not only as expressive as ReBAC but also it is able to detect implicit leaks of information as the one mentioned above. A formal comparison between the expressiveness of both frameworks is left as future work. The main advantage of ReBAC is its efficiency to enforce privacy policies, since it only requires to check whether the user who is trying to access some information is part of the audience. In our framework, we do not have performance results yet, hence we postpone the comparison to future work.

III.5.3 $\mathcal{FPPF}$ vs Epistemic Logic

The main difference between the semantics of $\mathcal{FPPF}$ and First-Order Epistemic Logic (FOEL) is the way knowledge is interpreted. SNMs "store" in each node a set of $\mathcal{F}_{\mathcal{KBL}}$ formulae that represent what an agent knows, namely the knowledge base of the agent. On the contrary, in relational Kripke structures, the *uncertainty* of the agents is modelled by means of a binary relation ($\mathcal{K}$) among states in the Kripke structure [29, 59]. The binary relation represents all the states that an agent considers possible. If a formula is true in all those states, then the agent knows that formula.

Nevertheless, this does not mean that the two models are complementary. In [29] Fagin *et al.* show how to construct *knowledge bases* for systems consisting of several agents by using *knowledge-based programming.* They define the state of an agent as a tuple containing all formulae the agent knows at a particular point in time. In addition to this information, the SNMs contain additional information regarding permissions and connections between users. As a matter of fact, we have shown that given a formula φ which characterises the knowledge, permission and connections of all agents in the SNM, a relational Kripke structure can be constructed containing the same information. Concretely, the canonical Kripke structure [29] resulting from φ can be built [72].

III.6 Conclusions and Future Work

In this paper we have presented a formal privacy policy framework which captures the dynamic behaviour of SNSs. The framework allows us to reason about privacy policies in dynamic social networks by means of a labelled transition system. The framework was enhanced with a set of operational semantics rules, which we instantiated for Twitter and for the tagging event in Facebook. We have shown how a designer can use our framework to model dynamic features of SNSs. Finally, we have introduced the notion of privacy-preserving SNS. As a proof-of-concept, we have formally proved that Twitter preserves privacy (according to our notion of privacy-preserving SNS). In addition, we have proved that adding new (and desirable) privacy policies to Twitter and Facebook makes their behaviour not privacy-preserving. We have also shown that the proofs provide useful information about which events are violating the privacy policies. In particular, we have shown how to update the Facebook instantiation to support new policies by analysing the information from the earlier proof where we showed that Facebook does not preserve the new privacy policy. In what follows we discuss some possible directions of future work.

Enforcement

There are two possible ways to make sure that an SNS is preserving-privacy using $\mathcal{FPPF}$. Firstly, designers can write a dynamic instantiation of the SNS that they want to implement. Then, they can formally prove that the operational semantics rules that were defined in that instantiation are privacy-preserving. This is similar to what we have shown for Twitter and Facebook. If the SNS designer proves that the SNS is privacy-preserving, then no verification at runtime is required, avoiding any additional overhead.

On the other hand, we would like to provide a runtime enforcement mechanism for SNSs under consideration. The main advantage of this approach is that it is partially independent of the implementation of the SNS. It only tracks the require information so that it can ensure that no privacy policy is violated. We are currently studying how to extract a monitor from the specification of the privacy policies, which would run in parallel with the SNS. The monitor checks that the privacy policies of the users are not violated as they execute events. To avoid the bottleneck of a centralised algorithm, we are considering a distributed implementation. We are already implementing $\mathcal{FPPF}$ in an open source SNS called Diaspora* [22, 21] to show the practicality of our approach.

Privacy Policies and Time

Privacy policies in $\mathcal{FPPF}$ cannot express real time properties. For example, a user may want to write a policies like *"My boss cannot know my location between 20:00-23:00"* or *"The audience of the post on my timeline during my birthday is only my friends"*. Adding a temporal component to our framework is a natural extension. Specific parts of the framework will become sensitive to the particular time at which the events happen. This needs to record when particular pieces of information are learnt, i.e., if Bob learnt Alice's location last week and today he learns it again, then then one should be able to tell apart these two locations.

In order to have a fine-grained control over time, we also need to differentiate between the timestamp of the information and when it was learnt. Imagine that Bob learns on Tuesday Alice's location from last Saturday. The predicate representing Alice's location has timestamp Saturday, but Bob learnt it on Tuesday. To make this distinction explicit, we can add timestamps to predicates and modalities. For example, the previous statement can be formalised as $K_{Bob}^{Tuesday} loc(Alice, Saturday)$. Additionally, we plan to include quantification over timestamps so that it is possible to specify intervals of time when privacy policies must be enforced.

III.A Appendix

III.A.1 Dynamic Instantiation of Twitter

In this appendix we provide a full dynamic instantiation for Twitter. We first provide the the set of events EVT_{Twitter}. Finally, we define the complete set of operational semantics rules for all of the events.

Set of events of Twitter

We define the set EVT_{Twitter} which contains all the events involved in the privacy analysis of Twitter.
EVT_{Twitter} consists of the following elements:

- *tweet* - It is one the core events of Twitter. It is used to post some piece information.
- *retweet* - It is used to share an already tweeted tweet.
- *favourite* - It allows users to classify tweets as favourite.
- *accessProf* - It represents the action of accessing a user's profile.
- *createProf* - It is the first event a user executes for joining Twitter. The user is required to provide a set of basic information which determines her profile.
- *follow* - Users can connect with other users by means of the Follower relationship.
- *acceptFollowReq* - When a user's profile is not public the *follow* event enables a request to the user. In order for the connection to be established the request must be accepted. This event represents the action of accepting the request.
- *block*, *unblock* - In Twitter a user can block other users. Not allowing to follow her, and can revert this decision.
- *showReco* - Twitter shows a selection of recommended-to-follow user recommendations to other users, when the email or the phone number of the recommended user is known by the one to whom the recommendation is shown.
- *showAdv* - This event models the action of a company sending an advertisement to a concrete user.
- *allowAdv*, *disallowAdv* - A user can (dis)allow a company from sending advertisement. These events model the activation and deactivation of this permission.

- *changeStPriv*, *changeStPub* - These events model the switching between *'Private'* or *'Public'* accounts.
- *inclLoc*, *notInclLoc* - These events represent whether the location is included or not in the tweet, respectively.

In what follows we provide the operational semantics rules for each of the events in EVT_{Twitter}.

Operational Semantics Rules of Twitter

Here we introduce all the operational semantics rules for the instantiation $\mathcal{FPPF}^{\mathcal{D}}_{\text{Twitter}}$. As usual, we divide them in Epistemic, Topological, Policy and Hybrid. Note that we only write the elements of $\mathcal{A}$ involved in the rule and we write "_" to denote the rest of the elements (see Definition 2).

Epistemic

R1 - Tweet

$$\frac{\begin{array}{c} Au = \mathit{followers}(tu) \cup \{u\} \cup \{u \mid \mathit{mention}(u, tu, \eta) \in \mathit{TweetInfo}\} \\ \mathit{state}(tu) == \text{'Public'} \\ \mathit{inclocation}(u) == \mathit{true} \qquad \forall p(\vec{t}) \in \mathit{TweetInfo}\ \forall i \in Au\ KB'(i) = KB(i) \cup \{C_{Au} p(\vec{t})\} \end{array}}{\langle \_, \_, KB, \_ \rangle \xrightarrow{\mathit{tweet}(tu, \mathit{TweetInfo})} \langle \_, \_, KB', \_ \rangle} \quad \text{(R1.1.1)}$$

$$\frac{\begin{array}{c} Au = \mathit{followers}(tu) \cup \{u\} \\ \mathit{state}(tu) == \text{'Private'} \qquad \mathit{inclocation}(u) == \mathit{false} \qquad \mathit{location}(tu, \eta) \notin \mathit{TweetInfo} \\ \forall p(\vec{t}) \in \mathit{TweetInfo}\ \forall i \in Au\ KB'(i) = KB(i) \cup \{C_{Au} p(\vec{t})\} \end{array}}{\langle \_, \_, KB, \_ \rangle \xrightarrow{\mathit{tweet}(tu, \mathit{TweetInfo})} \langle \_, \_, KB', \_ \rangle} \quad \text{(R1.2.2)}$$

$$\frac{\begin{array}{c} Au = \mathit{followers}(tu) \cup \{u\} \cup \{u \mid \mathit{mention}(u, tu, \eta) \in \mathit{TweetInfo}\} \\ \mathit{state}(tu) == \text{'Public'} \qquad \mathit{inclocation}(u) == \mathit{false} \qquad \mathit{location}(tu, \eta) \notin \mathit{TweetInfo} \\ \forall p(\vec{t}) \in \mathit{TweetInfo}\ \forall i \in Au\ KB'(i) = KB(i) \cup \{C_{Au} p(\vec{t})\} \end{array}}{\langle \_, \_, KB, \_ \rangle \xrightarrow{\mathit{tweet}(tu, \mathit{TweetInfo})} \langle \_, \_, KB', \_ \rangle} \quad \text{(R1.1.2)}$$

$$\frac{\begin{array}{c} Au = \mathit{followers}(tu) \cup \{u\} \qquad \mathit{state}(tu) == \text{'Private'} \\ \mathit{inclocation}(u) == \mathit{true} \qquad \forall p(\vec{t}) \in \mathit{TweetInfo}\ \forall i \in Au\ KB'(i) = KB(i) \cup \{C_{Au} p(\vec{t})\} \end{array}}{\langle \_, \_, KB, \_ \rangle \xrightarrow{\mathit{tweet}(tu, \mathit{TweetInfo})} \langle \_, \_, KB', \_ \rangle} \quad \text{(R1.2.1)}$$

R2 - Retweet

$$\frac{\begin{array}{c} F = \mathit{getTweetInfo}(tu, \eta) \qquad \mathit{state}(tu) == \text{'Public'} \\ \mathit{state}(rtu) == \text{'Public'} \qquad \mathit{TweetInfoAu} = \mathit{followers}(tu) \cup \mathit{followers}(rtu) \cup \{tu, rtu\} \\ \mathit{RetweetAu} = \mathit{followers}(tu) \cup \mathit{followers}(rtu) \cup \{tu, rtu\} \qquad \mathit{tweet}(tu, \eta) \in KB(rtu) \\ \forall p(\vec{t}) \in F\ \forall i \in \mathit{TweetInfoAu}\ KB'(i) = KB(i) \cup \{C_{Au} p(\vec{t})\} \\ \forall i \in \mathit{RetweetAu}\ KB'(i) = KB(i) \cup \{C_{\mathit{RetweetAu}} \mathit{retweet}(rtw, tu, \eta)\} \end{array}}{\langle \_, \_, KB, \_ \rangle \xrightarrow{\mathit{retweet}(rtu, \mathit{tweet}(tu, \eta))} \langle \_, \_, KB', \_ \rangle} \quad \text{(R2.1)}$$

$$\frac{\begin{array}{c} F = \mathit{getTweetInfo}(tu, \eta) \qquad \mathit{state}(tu) == \text{'Public'} \\ \mathit{state}(rtu) == \text{'Private'} \qquad \mathit{TweetInfoAu} = \mathit{followers}(tu) \cup \mathit{followers}(rtu) \cup \{tu, rtu\} \\ \mathit{RetweetAu} = \mathit{followers}(rtu) \cup \{rtu\} \qquad \mathit{tweet}(tu, \eta) \in KB(rtu) \\ \forall p(\vec{t}) \in F\ \forall i \in \mathit{TweetInfoAu}\ KB'(i) = KB(i) \cup \{C_{Au} p(\vec{t})\} \\ \forall i \in \mathit{RetweetAu}\ KB'(i) = KB(i) \cup \{C_{\mathit{RetweetAu}} \mathit{retweet}(rtw, tu, \eta)\} \end{array}}{\langle \_, \_, KB, \_ \rangle \xrightarrow{\mathit{retweet}(rtu, \mathit{tweet}(tu, \eta))} \langle \_, \_, KB', \_ \rangle} \quad \text{(R2.2)}$$

R3 - Favourite

$$\frac{tweet(tu,\eta) \in KB(fu) \qquad \forall i \in \{fu,tu\}\ KB'(i) = KB(i) \cup \{favourite(fu,tu,\eta)\}}{\langle\_,\_,KB,\_\rangle \xrightarrow{favourite(fu,tweet(tu,\eta))} \langle\_,\_,KB',\_\rangle} \quad \text{(R3)}$$

R4 - Access profile

$$\frac{\begin{array}{c} F = info(acd) \qquad [(acr,acd) \in A_{accessProf} \vee (acr,acd) \in A_{accessProfRec}] \\ state(acd) = \text{'Public'} \qquad \forall p(\overrightarrow{t}) \in F\ KB'(acr) = KB(acr) \cup \{p(\overrightarrow{t})\} \end{array}}{\langle\_,\{\{A_i\}_{i\in\Sigma},\_\},KB,\_\rangle \xrightarrow{accessProf(acr,acd)} \langle\_,\{\{A_i\}_{i\in\Sigma},\_\},KB',\_\rangle} \quad \text{(R4.1)}$$

$$\frac{\begin{array}{c} F = info(acd) \\ [(acr,acd) \in A_{accessProf} \vee (acr,acd) \in A_{accessProfRec}] \qquad state(acd) = \text{'Private'} \\ (acd,acr) \in C_{Follower} \qquad \forall p(\overrightarrow{t}) \in F\ KB'(acr) = KB(acr) \cup \{p(\overrightarrow{t})\} \end{array}}{\langle\_,\{\{A_i\}_{i\in\Sigma},\{\{C_i\}_{i\in\mathcal{C}},\_\},KB,\_\rangle \xrightarrow{accessProf(acr,acd)} \langle\_,\{\{A_i\}_{i\in\Sigma},\{C_i\}_{i\in\mathcal{C}},\_\}KB',\_\rangle} \quad \text{(R4.2)}$$

R10 - Show recommendation

$$\frac{\begin{array}{c} beingReco(recommended) == false \qquad email(recommended) \in KB(viewer) \\ A'_{accessProfRec} = A_{accessProfRec} \cup \{(viewer, recommended)\} \end{array}}{\langle\_,\{\{A_i\}_{i\in\Sigma},\_\},KB,\_\rangle \xrightarrow{showReco(recommended,viewer)} \langle\_,\{\{A'_i\}_{i\in\Sigma},\_\},KB,\_\rangle} \quad \text{(R10.1)}$$

$$\frac{\begin{array}{c} beingReco(recommended) == true \\ A'_{accessProfRec} = A_{accessProfRec} \cup \{(viewer, recommended)\} \end{array}}{\langle\_,\{\{A_i\}_{i\in\Sigma},\_\},\_,\_\rangle \xrightarrow{showReco(recommended,viewer)} \langle\_,\{\{A'_i\}_{i\in\Sigma},\_\},\_,\_\rangle} \quad \text{(R10.2)}$$

R11 - Show advertisment

$$\frac{(advertiser, user) \in A_{sendAd} \qquad KB'(user) = KB(user) \cup \{advertise(advertiser,\eta)\}}{\langle\_,\{\{A_i\}_{i\in\Sigma},\_\},KB,\_\rangle \xrightarrow{showAdv(advertiser,user)} \langle\_,\{\{A_i\}_{i\in\Sigma},\_\},KB',\_\rangle} \quad \text{(R11)}$$

Topological

R6 - Follow (R6.2 is a hybrid rule)

$$\frac{\begin{array}{c} (followed, follower) \notin C_{Block} \qquad state(followed) == \text{'Public'} \\ (follower, followed) \notin C_{Follower} \qquad C'_{Followers} = C_{Followers} \cup \{(follower, followed)\} \end{array}}{\langle\_,\{\{C_i\}_{i\in\mathcal{C}},\_\},\_,\_\rangle \xrightarrow{follow(follower,followed)} \langle\_,\{\{C'_i\}_{i\in\mathcal{C}},\_\},\_,\_\rangle} \quad \text{(R6.1)}$$

R7 - Accept follow request

$$\frac{\begin{array}{c} \mathit{followRequest}(\mathit{accepted}) \in \mathit{KB}((\mathit{accepter}) \\ C'_{\mathit{Followers}} = C_{\mathit{Followers}} \cup \{(\mathit{follower}, \mathit{followed})\} \end{array}}{\langle \_, \{\{C_i\}_{i\in\mathcal{C}}, \_\}, \mathit{KB}, \_\rangle \xrightarrow{(\mathit{acceptFollowReq}((\mathit{accepter},(\mathit{accepted})} \langle \_, \{\{C'_i\}_{i\in\mathcal{C}}, \_\}, \mathit{KB}, \_\rangle} \quad \text{(R7)}$$

R8 - Block

$$\frac{\begin{array}{c} (\mathit{blocker}, \mathit{blocked}) \notin C_{\mathit{Follower}} \\ (\mathit{blocker}, \mathit{blocked}) \notin C_{\mathit{Block}} \qquad C'_{\mathit{Block}} = C_{\mathit{Block}} \cup \{(\mathit{blocker}, \mathit{blocked})\} \end{array}}{\langle \_, \{\{C_i\}_{i\in\mathcal{C}}, \_\}, \_, \_\rangle \xrightarrow{\mathit{block}(\mathit{blocker},\mathit{blocked})} \langle \_, \{\{C'_i\}_{i\in\mathcal{C}}, \_\}, \_, \_\rangle} \quad \text{(R8.1)}$$

$$\frac{\begin{array}{c} (\mathit{blocker}, \mathit{blocked}) \in C_{\mathit{Follower}} \qquad (\mathit{blocker}, \mathit{blocked}) \notin C_{\mathit{Block}} \\ C'_{\mathit{Block}} = C_{\mathit{Block}} \cup \{(\mathit{blocker}, \mathit{blocked})\} \qquad C'_{\mathit{Followers}} = C_{\mathit{Followers}} \setminus \{(\mathit{blocker}, \mathit{blocked})\} \end{array}}{\langle \_, \{\{C_i\}_{i\in\mathcal{C}}, \_\}, \_, \_\rangle \xrightarrow{\mathit{block}(\mathit{blocker},\mathit{blocked})} \langle \_, \{\{C'_i\}_{i\in\mathcal{C}}, \_\}, \_, \_\rangle} \quad \text{R8.2}$$

R9 - Unblock

$$\frac{(\mathit{unblocker}, \mathit{unblocked}) \in R_{\mathit{Block}} \qquad C'_{\mathit{Block}} = C_{\mathit{Block}} \setminus \{(\mathit{unblocker}, \mathit{unblocked})\}}{\langle \_, \{\{C_i\}_{i\in\mathcal{C}}, \_\}, \_, \_\rangle \xrightarrow{\mathit{unblock}(\mathit{unblocker},\mathit{unblocked})} \langle \_, \{\{C'_i\}_{i\in\mathcal{C}}, \_\}, \_, \_\rangle} \quad \text{(R9)}$$

Policy

R14 - Change state to private

$$\frac{\pi'_u = \pi_u \cup \{P1(u), P2(u)\}}{\langle \_, \_, \_, \pi\rangle \xrightarrow{\mathit{changeStPriv}(u)} \langle \_, \_, \_, \pi'\rangle} \quad \text{(R14)}$$

R15 - Change state to public

$$\frac{\pi'_u = \pi_u \setminus \{P1(u), P2(u)\}}{\langle \_, \_, \_, \pi\rangle \xrightarrow{\mathit{changeStPub}(u)} \langle \_, \_, \_, \pi'\rangle} \quad \text{(R15)}$$

R16 - Include location on Tweets

$$\frac{\pi'_u = \pi_u \setminus \{P3(u)\}}{\langle \_, \_, \_, \pi\rangle \xrightarrow{\mathit{inclLoc}(u)} \langle \_, \_, \_, \pi'\rangle} \quad \text{(R16)}$$

R17 - Not include location on Tweets

$$\frac{\pi'_u = \pi_u \cup \{P3(u)\}}{\langle \_, \_, \_, \pi\rangle \xrightarrow{\mathit{notInclLoc}(u)} \langle \_, \_, \_, \pi'\rangle} \quad \text{(R17)}$$

Hybrid

R5 - Create profile

$$\frac{\begin{array}{c} u \notin Ag \qquad Ag' = Ag \cup \{u\} \qquad KB'_i = \mathit{InitialInfo} \\ \forall j \in Advertisers\ A'_{sendAd} = A_{sendAd} \cup \{(u,j)\} \qquad A'_{accessProf} = A_{accessProf} \cup \{(u,u)\} \end{array}}{\langle Ag, \{\{A_i\}_{i\in\Sigma}, \_\}, KB, \_\rangle \xrightarrow{createProf(u, \mathit{InitialInfo})} \langle Ag', \{\{A'_i\}_{i\in\Sigma}, \_\}, KB', \_\rangle} \tag{R5}$$

R6 - Follow (R6.1 is a topological rule)

$$\frac{\begin{array}{c} (\mathit{followed}, \mathit{follower}) \notin C_{Block} \\ \mathit{state}(\mathit{followed}) == \text{'}\mathit{Private}\text{'} \qquad (\mathit{follower}, \mathit{followed}) \notin C_{Follower} \\ \mathit{Request} = \{C_{\{\mathit{followed}, \mathit{follower}\}} followRequest(\mathit{follower})\} \\ \forall i \in \{\mathit{followed}, \mathit{follower}\}\ KB'(i) = KB(i) \cup \mathit{Request} \end{array}}{\langle \_, \{\{C_i\}_{i\in\mathcal{C}}, \_\}, KB, \_\rangle \xrightarrow{\mathit{follow}(\mathit{follower}, \mathit{followed})} \langle \_, \{\{C_i\}_{i\in\mathcal{C}}, \_\}, KB', \_\rangle} \tag{R6.2}$$

R12 - Allow advertisment

$$\frac{\begin{array}{c} \forall i \in Advertisers\ A'_{sendAd} = A_{sendAd} \cup \{(i,u)\} \\ \pi'_u = \pi_u \cup \{P5(u)\} \end{array}}{\langle \_, \{\{A_i\}_{i\in\Sigma}, \_\}, \_, \pi\rangle \xrightarrow{allowAdv(Advertisers, u)} \langle \_, \{\{A'_i\}_{i\in\Sigma}, \_\}, \_, \pi'\rangle} \tag{R12}$$

R13 - Disallow advertisement

$$\frac{\begin{array}{c} P5(u) \in \pi(u) \qquad \forall i \in Advertisers\ A'_{sendAd} = A_{sendAd} \setminus \{(i,u)\} \\ \pi'_u = \pi_u \setminus \{P5(u)\} \end{array}}{\langle \_, \{\{A_i\}_{i\in\Sigma}, \_\}, \_, \pi\rangle \xrightarrow{disallowAdv(Advertisers, u)} \langle \_, \{\{A'_i\}_{i\in\Sigma}, \_\}, \_, \pi'\rangle} \tag{R13}$$

III.A.2 Proofs

Theorem 1 - Twitter is privacy-preserving

The proof will be split in as many cases as rules we defined for $\mathcal{FPPF}^{\mathcal{D}}_{\text{Twitter}}$, i.e. from $R1$ to $R17$, where we show that any rule will violate any privacy policy. For each of the rules we will state which privacy policies could be violated. The structure of each case of the proof is similar. We proof for all the policies that could violate the event that if the privacy policy is in conformance with the SNM before the execution of the event, then after the execution of the event, the privacy policy is still preserved in the resulting SNM. We start by assuming that after the executing of the event the policy is violated and later we show that it leads to a contradiction. After proving it for all for rules and privacy policies we conclude that Twitter is privacy-preserving. In the proof we use **bold** text to state the rule and the possible privacy policies which it can violate, and underline text to split the proof cases for each of those privacy policies.

Proof. **$R1$ – The execution of $R1$ could only violate the policies $P1$ and $P3$**

9. Executing $R1$ and $P1$ enabled

9.1. Given
9.1.1. $u \in Ag$ (owner of the privacy policy $P1(u)$)
9.1.2. Predicates to be disclosed $\mathit{TweetInfo} \subseteq 2^{\mathcal{P}}$ where $\mathit{tweet}(u, \eta) \in \mathit{TweetInfo}$
9.1.3. $e = \mathit{tweet}(u, \mathit{TweetInfo})$
9.1.4. We want to prove:

$$SN \vDash_C P1(u) \text{ and } SN \xrightarrow{e} SN' \text{ then } SN' \vDash_C P1(u)$$

9.2. By contradiction, let us assume
9.2.1. $SN \vDash_C P1(u)$ and $SN \xrightarrow{e} SN'$
9.2.2. $SN' \nvDash_C P1(u)$

9.3. By 18.2.2.
9.3.1. $SN' \nvDash_C P1(u)$ [Definition $\vDash_C$]
9.3.2. $SN', u \vDash \neg\neg S_{Ag \setminus followers(u) \setminus \{u\}} \mathit{tweet}(u, \eta)$ $[\neg\neg_e]$
9.3.3. $SN', u \vDash S_{Ag \setminus followers(u) \setminus \{u\}} \mathit{tweet}(u, \eta)$

9.4. By 18.3.5. and the definition of $\vDash$ we have
9.4.1. $\exists i \in Ag \setminus \mathit{followers}(u) \setminus \{u\}$ s.t. $SN' \vDash K_i tweet(u, \eta)$

9.5. By Definition of $R1$, we have that
9.5.1. $\forall p(\overrightarrow{t}) \in \mathit{TweetInfo}\ SN' \vDash C_{followers(u) \cup \{u\}} p(\overrightarrow{t})$ [By 9.1.2.]
9.5.2. $SN' \vDash C_{followers(u) \cup \{u\}} tweet(u, \eta)$ [By $\vDash$]
9.5.3. $SN' \vDash E^0_{followers(u) \cup \{u\}} tweet(u, \eta) \wedge$
$E^1_{followers(u) \cup \{u\}} tweet(u, \eta) \wedge$
$E^2_{followers(u) \cup \{u\}} tweet(u, \eta) \wedge$
$E^3_{followers(u) \cup \{u\}} tweet(u, \eta) \wedge \ldots$ [By $\vDash$]
9.5.4. $SN' \vDash E^1_{followers(u) \cup \{u\}} tweet(u, \eta)$ [By $\vDash$]
9.5.5. $\forall j \in \mathit{followers}(u) \cup \{u\}\ SN \vDash K_j tweet(u, \eta)$

9.6. By 18.2.1. we have
9.6.1. $SN \vDash_C P1(u)$ [By $\vDash_C$]
9.6.2. $SN \vDash \neg S_{Ag \setminus followers(u) \setminus \{u\}} tweet(u, \eta)$ [By Definition S_G]
9.6.3. $SN \vDash \neg(\bigvee_{i \in Ag \setminus followers(u) \setminus \{u\}} K_i tweet(u, \eta))$ [Morgan]
9.6.4. $SN \vDash \bigwedge_{i \in Ag \setminus followers(u) \setminus \{u\}} \neg K_i tweet(u, \eta)$

9.7. By 9.6.4. and 18.5.4. we have
9.7.1. $SN' \vDash_C P1(u)$

9.8. By 18.2.2. and 9.7.1. we derive a contradiction. $\square$

10. Executing $R1$ and $P3$ enabled

10.1. Given
10.1.1. $u \in Ag$ (owner of the privacy policy $P3(u)$)
10.1.2. Predicates to be disclosed $\mathit{TweetInfo} \subseteq 2^{\mathcal{P}}$
10.1.3. Location of the tweet $\mathit{location}(u, \eta)$
10.1.4. $Au \subseteq Ag$
10.1.5. $e = tweet(u, \mathit{TweetInfo})$
10.1.6. We want to prove:

$$SN \vDash_C P3(u) \text{ and } SN \xrightarrow{e} SN' \text{ then } SN' \vDash_C P3(u)$$

10.2. By contradiction, let us assume
10.2.1. $SN \vDash_C P3(u)$ and $SN \xrightarrow{e} SN'$
10.2.2. $SN' \nvDash_C P3(u)$

10.3. By 10.2.1. and $\vDash_C$
10.3.1. $SN' \vDash \neg\neg S_{Ag\setminus\{u\}} location(u, \eta)$ $[\neg\neg_e]$
10.3.2. $SN' \vDash S_{Ag\setminus\{u\}} location(u, \eta)$ [By $\vDash$]
10.3.3. $\exists i \in Ag \setminus \{u\}$ such that $SN' \vDash K_i location(u, \eta)$

10.4. By Definition of $R1$
10.4.1. $\forall p(\overrightarrow{t}) \in TweetInfo \setminus \{location(u, \eta)\}$ $SN' \vDash C_{Au} p(\overrightarrow{t})$

10.5. By 10.2.1. and the definition of $\vDash_C$
10.5.1. $SN \vDash \neg S_{Ag\setminus\{u\}} location_\eta$ [Definition S_G]
10.5.2. $SN \vDash \neg(\bigvee_{i\in Ag\setminus\{u\}} K_i location(u, \eta))$ [Morgan]
10.5.3. $SN \vDash \bigwedge_{i\in Ag\setminus\{u\}} \neg K_i location(u, \eta)$

10.6. By 10.4.1. and 10.5.3.
10.6.1. $SN' \vDash_C P3(u)$

10.7. By 10.6.1. and 10.2.2. we derive a contradiction. $\square$

$R2$ - The execution of $R2$ could only violate the policies $P2$ and $P3$

11. <u>Executing $R2$ and $P2$ enabled</u>

11.1. Given
11.1.1. $u \in Ag$ (owner of $P2(u)$ and retweeter)
11.1.2. $tweet(tu, \eta)$ (tweet $\eta \in \mathbb{N}$ of user $tu \in Ag$)
11.1.3. $e = retweet(u, tweet(tu, \eta))$
11.1.4. We want to prove:

$$SN \vDash_C P2(u) \text{ and } SN \xrightarrow{e} SN' \text{ then } SN' \vDash_C P2(u)$$

11.2. By contradiction, let us assume
11.2.1. $SN \vDash_C P2(u)$ and $SN \xrightarrow{e} SN'$
11.2.2. $SN' \nvDash_C P2(u)$

11.3. By 11.2.1. and $\vDash_C$
11.3.1. $SN' \vDash \neg\neg S_{Ag \setminus followers(u) \setminus \{u\}} retweet(u, tu, \eta)$ $[\neg\neg_e]$
11.3.2. $SN' \vDash S_{Ag \setminus followers(u) \setminus \{u\}} retweet(u, tu, \eta)$ [By $\vDash$]
11.3.3. $\exists i \in Ag \setminus followers(u) \setminus \{u\}$ such that $SN' \vDash K_i location(u, \eta)$

11.4. By Definition of $R2$
11.4.1. $SN' \vDash C_{followers(u) \cup \{u\}} retweet(u, tu, \eta)$ [By $\vDash$]
11.4.2. $SN' \vDash C_{followers(u) \cup \{u\}} retweet(u, tu, \eta)$ [By $\vDash$]
11.4.3. $SN' \vDash E^0_{followers(u) \cup \{u\}} retweet(u, tu, \eta) \wedge$
$E^1_{followers(u) \cup \{u\}} retweet(u, tu, \eta) \wedge$
$E^2_{followers(u) \cup \{u\}} retweet(u, tu, \eta) \wedge$
$E^3_{followers(u) \cup \{u\}} retweet(u, tu, \eta) \wedge \ldots$ [By $\vDash$]
11.4.4. $SN' \vDash E^1_{followers(u) \cup \{u\}} retweet(u, tu, \eta)$ [By $\vDash$]
11.4.5. $\forall j \in followers(u) \cup \{u\}$ $SN', j \vDash K_j retweet(u, tu, \eta)$

11.5. By 11.2.1. and the definition of $\vDash_C$
11.5.1. $SN \vDash \neg S_{Ag \setminus followers(u) \setminus \{u\}} retweet(u, tu, \eta)$ [Definition S_G]
11.5.2. $SN \vDash \neg(\bigvee_{i \in Ag \setminus followers(u) \setminus \{u\}} K_i retweet(u, tu, \eta))$ [Morgan]
11.5.3. $SN \vDash \bigwedge_{i \in Ag \setminus followers(u) \setminus \{u\}} \neg K_i retweet(u, tu, \eta)$

11.6. By 11.4.5. and 11.5.3.
11.6.1. $SN' \vDash_C P2(u)$

11.7. By 11.6.1. and 11.2.2. we derive a contradiction. □

12. <u>Executing $R2$ and $P3$ enabled</u>

12.1. Given
12.1.1. $u \in Ag$ (owner of $P3(u)$ and retweeter)
12.1.2. $tweet(tu, \eta)$ (tweet $\eta \in \mathbb{N}$ of user $tu \in Ag$)

12.1.3. $\mathit{TweetInfoAu} \subseteq Ag$ (audience of the retweeted tweet)
12.1.4. $\mathit{RetweetAu} \subseteq Ag$ (audience of the fact of retweeting)
12.1.5. $e = \mathit{retweet}(u, \mathit{tweet}(tu, \eta))$
12.1.6. We want to prove:

$$\mathit{SN} \vDash_C P3(u) \text{ and } \mathit{SN} \xrightarrow{e} \mathit{SN}' \text{ then } \mathit{SN}' \vDash_C P3(u)$$

12.2. By contradiction, let us assume
12.2.1. $\mathit{SN} \vDash_C P3(u)$ and $\mathit{SN} \xrightarrow{e} \mathit{SN}'$
12.2.2. $\mathit{SN}' \nvDash_C P3(u)$

12.3. By 12.2.1. and $\vDash_C$
12.3.1. $\mathit{SN}' \vDash \neg\neg S_{Ag\setminus\{u\}} \mathit{location}(tu, \eta)$ $[\neg\neg_e]$
12.3.2. $\mathit{SN}' \vDash S_{Ag\setminus\{u\}} \mathit{location}(tu, \eta)$ [By $\vDash$]
12.3.3. $\exists i \in Ag \setminus \{u\}$ such that $\mathit{SN}' \vDash K_i \mathit{location}(u, \eta)$

12.4. By Definition of $R2$
12.4.1. $\forall p(\overrightarrow{t}) \in \mathit{getTweetInfo}(tu, \eta) \setminus \{\mathit{location}(tu, \eta)\}$ $\mathit{SN}' \vDash C_{\mathit{TweetInfoAu}} p(\overrightarrow{t})$

12.5. By 12.2.1. and the definition of $\vDash_C$
12.5.1. $\mathit{SN} \vDash \neg S_{Ag\setminus\{u\}} \mathit{location}(tu, \eta)$ [Definition S_G]
12.5.2. $\mathit{SN} \vDash \neg(\bigvee_{i\in Ag\setminus\{u\}} K_i \mathit{location}(tu, \eta))$ [Morgan]
12.5.3. $\mathit{SN} \vDash \bigwedge_{i\in Ag\setminus\{u\}} \neg K_i \mathit{location}(tu, \eta)$

12.6. By 12.4.1. and 12.5.3.
12.6.1. $\mathit{SN}' \vDash_C P3(u)$

12.7. By 12.6.1. and 12.2.2. we derive a contradiction. □

***R*3 – None of the privacy policies in Definition 12 can be violated by rule *R*3**

Since $\mathit{favourite}(u, tu, \eta)$ is the only predicate disclosed during the execution of $R3$ and none of the privacy policies in Definition 12 specify any restriction against this predicate, it would not be possible that a violation of them occurs.

*R*4 – The execution of *R*4 could violate the privacy policies *P*1, *P*2, *P*3

13. Executing $R4$ and $P1$ enabled

13.1. Given
13.1.1. $acd \in Ag$ (owner of $P1(acd)$)
13.1.2. $acr \in Ag$ (agent who is executing $R4$)
13.1.3. $e = accessProf(acd, acr)$
13.1.4. We want to prove:

$$SN \vDash_C P1(acd) \text{ and } SN \xrightarrow{e} SN' \text{ then } SN' \vDash_C P1(acd)$$

13.2. We assume
13.2.1. $\exists\, tweet(acd, \eta) \in info(acd)$ (otherwise 13.1.4. trivially holds)

13.3. By contradiction, let us assume
13.3.1. $SN \vDash_C P1(acd)$ and $SN \xrightarrow{e} SN'$
13.3.2. $SN' \nvDash_C P1(acd)$

13.4. By 13.3.2.
13.4.1. $SN' \nvDash_C P1(acd)$ [Definition $\vDash_C$]
13.4.2. $SN' \vDash \neg\neg S_{Ag\setminus followers(u)\setminus\{u\}} u.tweet_\eta$ [$\neg\neg_e$]
13.4.3. $SN' \vDash S_{Ag\setminus followers(u)\setminus\{u\}} tweet(u, \eta)$[By $\vDash$]
13.4.4. $\exists i \in Ag \setminus followers(u) \setminus \{u\}$ s.t. $SN' \vDash K_i tweet(acd, \eta)$

13.5. By 13.3.1., 13.1.3., Definition of *state* and Definition of $R4$, we have that
13.5.1. If $(acr, acd) \in C_{Follower}$
13.5.1.1. $\forall p(\overrightarrow{t}) \in info(acd)\ SN' \vDash K_{acr} p(\overrightarrow{t})$ [By 13.2.1.]
13.5.1.2. $SN', acr \vDash K_{acr} tweet(acd, \eta)$

13.5.2. If $(acr, acd) \notin C_{Follower}$
13.5.2.1. $R4$ will not be executed.

13.6. By 13.3.1. we have
13.6.1. $SN \models_C P1$ [By $\models_C$]
13.6.2. $SN \models \neg S_{Ag \setminus followers(acd) \setminus \{acd\}} tweet(acd, \eta)$ [By Definition S_G]
13.6.3. $SN \models \neg(\bigvee_{i \in Ag \setminus followers(acd) \setminus \{acd\}} K_i tweet(acd, \eta))$ [Morgan]
13.6.4. $SN \models \bigwedge_{i \in Ag \setminus followers(acd) \setminus \{acd\}} \neg K_i tweet(acd, \eta)$

13.7. By 13.6.4. and 13.5.1.2. and 13.5.2.1. we have
13.7.1. $SN' \models_C P1(acd)$

13.8. By 13.3.2. and 13.7.1. we derive a contradiction. □

14. Executing $R4$ and $P2$ or $P3$ enabled

14.1. The exact same reasoning as before can be applied for $P2$ and $P3$ by replacing $tweet(acd, \eta)$ with $retweet(acd, u, \eta)$ or $location(acd, \eta)$, respectively. This is because the function $info(acd)$ will return also those predicates in case the are part of the accessed user information.

$R5 - R9$ – None of the privacy policies in Definition 12 can be violated by the rules $R5 - R9$
Since there is neither disclosure of information nor granting of permission it is not possible to violate any of the defined privacy policies.

$R10$ – The execution of $R10$ could violate the privacy policy $P4$

15. Executing $R10$ and $P4$ enabled

15.1. Given
15.1.1. $r \in Ag$ (Owner of $P4(r)$, i.e. $P4(r) \in \pi_r$)

15.1.2. If $P4(r) \in \pi_r$ then the case $R10.1$ is the one which will be executed
15.1.3. $e = \mathit{showReco}(r, v)$
15.1.4. We want to prove:

$$SN \vDash_C P4(r) \text{ and } SN \xrightarrow{e} SN' \text{ then } SN' \vDash_C P4(r)$$

15.2. By contradiction, let us assume
15.2.1. $SN \vDash_C P4(r)$ and $SN \xrightarrow{e} SN'$
15.2.2. $SN' \nvDash_C P4(r)$

15.3. By 15.2.2.
15.3.1. $SN' \nvDash_C P4(r)$ [By $\vDash_C$]
15.3.2. $SN' \vDash \neg\forall x.(\neg K_x(\mathit{email}(r) \vee \mathit{phone}(r)) \implies \neg P_x^r \mathit{accessProfRec})$ [By $\neg\forall z.\varphi \equiv \exists z.\neg\varphi$]
15.3.3. $SN' \vDash \exists x.\neg(\neg K_x(\mathit{email}(r) \vee \mathit{phone}(r)) \implies \neg P_x^r \mathit{accessProfRec})$ [By $\exists_e$, where $\varphi[v/x]$]
15.3.4. $SN' \vDash \neg(\neg K_v(\mathit{email}(r) \vee \mathit{phone}(r)) \implies \neg P_v^r \mathit{accessProfRec})$ [By $\vDash$]
15.3.5. $SN' \vDash \neg(\neg(\neg K_v(\mathit{email}(r) \vee \mathit{phone}(r))) \vee (\neg P_v^r \mathit{accessProfRec}))$ [$\neg\neg_e$]
15.3.6. $SN' \vDash \neg(K_v(\mathit{email}(r) \vee \mathit{phone}(r)) \vee \neg P_v^r \mathit{accessProfRec})$ [Morgan]
15.3.7. $SN' \vDash \neg K_v(\mathit{email}(r) \vee \mathit{phone}(r)) \wedge P_v^r \mathit{accessProfRec}$

15.4. By 15.1.2. and 15.2.1. and Definition of $R10$
15.4.1. If $SN \vDash K_v(\mathit{email}(r) \vee \mathit{phone}(r))$
15.4.1.1. $SN' \vDash K_v(\mathit{email}(r) \vee \mathit{phone}(r)) \wedge P_v^r acessProfRecommended$
15.4.2. If $SN \vDash \neg K_v(\mathit{email}(r) \vee \mathit{phone}(r))$
15.4.2.1. $R10.1$ is not executed

15.5. By 15.2.1. we have
15.5.1. $SN \vDash_C P4(r)$ [By $\vDash_C$]
15.5.2. $SN \vDash \neg K_v(\mathit{email}(r) \vee \mathit{phone}(r)) \implies \neg P_v^r \mathit{accessProfRec}$ [By $\vDash$]
15.5.3. $SN \vDash \neg\neg K_v(\mathit{email}(r) \vee \mathit{phone}(r)) \vee \neg P_v^r \mathit{accessProfRec}$ [$\neg\neg_e$]
15.5.4. $SN \vDash K_v(\mathit{email}(r) \vee \mathit{phone}(r)) \vee \neg P_v^r \mathit{accessProfRec}$ [$\neg\neg_i$]
15.5.5. $SN \vDash \neg\neg(K_v(\mathit{email}(r) \vee \mathit{phone}(r)) \vee \neg P_v^r \mathit{accessProfRec})$ [Morgan]
15.5.6. $SN \vDash \neg(\neg K_v(\mathit{email}(r) \vee \mathit{phone}(r)) \wedge P_v^r \mathit{accessProfRec})$

15.6. By 15.5.6. and 15.4.1.1.
15.6.1. $SN' \vDash_C P4(r)$

15.7. By 15.6.1. and 15.2.2. we derive a contradiction. □

$R11 - R13$ – None of the privacy policies in Definition 12 can be violated by the rules $R11 - R13$

One could think that $R11$ may violate $P5$. However, the policy is preserved intrinsically in the definition of the rules $R12, R13$. Since it is checked beforehand if an advertiser have permission or not to send an advertisement. Basically activating or deactivating the policy would mean granting or removing permission to the advertisement companies to execute the action *sendAd* to the user.

$R14 - R17$ – None of the privacy policies in Definition 12 can be violated by the rules $R14 - R17$

These rules only aggregate or remove privacy policies to the users, they don't modify neither their knowledge nor their permission.

Finally we can conclude that $\mathcal{FPPF}^{\mathcal{D}}_{\text{Twitter}}$ is a privacy-preserving social network. □

Lemma 1 - Twitter is not privacy-preserving

We will show that from a social network model which preserves the privacy policy, after executing the event *tweet* (as it is defined in $\mathcal{FPPF}^{\mathcal{D}}_{\text{Twitter}}$) mentioning a user and adding the location, the privacy policy would be violated.

Proof Sketch: Assume a user $u \in Ag$ who has never been mentioned and has one instance of $P6(u)$ in her set of policies, and another user $o \in Ag$ who executes the event

$$e = tweet(o, \{tweet(o, \eta), mention(u, o, \eta), location(o, \eta)\}).$$

If the result of executing the event in SN is SN', $SN \xrightarrow{e} SN'$, then by assumption we know that $SN \vDash_C P6(u)$, but according to $R1$, we know that all users in the audience of the tweet will learn $mention(u, o, \eta)$ and $location(o, \eta)$ and therefore $SN' \nvDash_C P6(u)$.

Proof.

16. Executing R1 and P6 activated

16.1. Given
16.1.1. User $u \in Ag$ such that $SN \vDash_C P6(u)$
16.1.2. $inclocation(u) == true$
16.1.3. $TweetInfo = \{tweet(tu, \eta), location(tu, \eta), mention(u, tu, \eta)\}$
16.1.4. $e = tweet(tu,\ TweetInfo)$
16.1.5. We want to prove

$$SN \vDash_C P6(u) \text{ and } SN \xrightarrow{e} SN' \text{ then } SN' \nvDash_C P6(u)$$

16.2. Let us assume
16.2.1. $SN' \nvDash_C P6(u)$ [By $\vDash_C$]
16.2.2. $SN' \vDash \neg(K_i location(tu, \eta) \wedge K_i mention(u, tu, \eta))$

16.3. Let us assume
16.3.1. $SN \vDash_C P6(u)$ and $SN \xrightarrow{e} SN'$

16.4. By the Definition of $R1$ and 17.1.7.
16.4.1. If $state(tu) ==$ *'Public'*
16.4.2. $\forall p(\vec{t}) \in TweetInfo\ SN' \vDash C_{followers(tu) \cup \{tu\}} p(\vec{t})$ [By 16.1.3.]
16.4.3. $SN' \vDash C_{followers(tu) \cup \{tu\}} location(tu, \eta) \wedge mention(u, tu, \eta)$ [By $\vDash$]
16.4.4. $\forall i \in followers(tu) \cup \{tu\} SN' \vDash K_i location(tu, \eta) \wedge mention(u, tu, \eta)$
16.4.5. If $state(tu) ==$ *'Private'*
16.4.6. $\forall p(\vec{t}) \in TweetInfo\ SN' \vDash C_{followers(tu) \cup \{tu\} \cup \{u\}} p(\vec{t})$ [By 16.1.3.]
16.4.7. $SN' \vDash C_{followers(tu) \cup \{tu\} \cup \{u\}} location(tu, \eta) \wedge mention(u, tu, \eta)$ [By $\vDash$]
16.4.8. $\forall i \in followers(tu) \cup \{tu\} \cup \{u\} SN' \vDash K_i location(tu, \eta) \wedge mention(u, tu, \eta)$

16.5. By 16.4.4. and 16.4.8.
16.5.1. $\exists i \in followers(tu) \cup \{tu\} SN' \vDash K_i location(tu, \eta) \wedge mention(u, tu, \eta)$

16.6. By 16.5.1. and 16.2.2. we derive a contradiction.

$\square$

Lemma 2 - Facebook is not privacy-preserving

We will show that from a social network model which preserves the privacy policy $FP1$, after executing the event *acceptTagRequest* (as it is defined in $\mathcal{FPPF}^{\mathcal{D}}_{\text{FB-Tag}}$) a user can be tagged without approving herself the tag.

Proof sketch: Let $tge \in Ag$ be a user who has never been tagged and let $tgr \in Ag$ be a user who has executed the event $tag(tgr, tge, o, \eta)$ in order to tag tge in $picture(o, \eta)$ where $o \in Ag$ and $\eta \in \mathbb{N}$. The owner of $picture(o, \eta)$ is o. Therefore, in the current social network model SN, it holds that $tagRequest(tge, tgr, o, \eta) \in KB(o)$. In order for $\mathcal{FPPF}^{\mathcal{D}}_{\text{FB-Tag}}$ to preserve privacy it must hold that if $SN \models_C FP1(tge)$ and $SN \xrightarrow{acceptTagRequest(o,tge,tgr,picture(o,\eta))} SN'$ where $SN, SN' \in \mathcal{SN}_{\text{FB-Tag}}$ then $SN' \models_C FP1(tge)$.

Since tge was not tagged before the execution of $FR2$ we know that $SN \models_C FP1(tge)$. Also since $tagRequest(tge, tgr, o, \eta) \in KB(o)$ and $acptr == o$ we know that $FR2$ can be executed. By the definition of $FR2$, we know that $SN' \models E_{Au}tag(tge, o, o, \eta)$, hence $SN' \not\models_C FP1(tge)$, which contradicts our claim $SN' \models_C FP1(tge)$ and therefore $\mathcal{FPPF}^{\mathcal{D}}_{\text{FB-Tag}}$ is not privacy-preserving.

Proof.

17. Executing FR1 and FP1 activated

17.1. Given
17.1.1. User $tge \in Ag$ such that $SN \models_C FP1(tge)$
17.1.2. User $o \in Ag$ such that $tge \,!= o$
17.1.3. Picture $picture(o, \eta)$ where $\eta \in \mathbb{N}$
17.1.4. User $tgr \in Ag$
17.1.5. The owner of the picture is part of its audience $o \in Au$
17.1.6. $Au = audience(picture(o, \eta))$
17.1.7. $e = acceptTagRequest(o, tge, tgr, picture(o, \eta))$
17.1.8. We want to prove

$$SN \models_C FP1(u) \text{ and } SN \xrightarrow{e} SN' \text{ then } SN' \not\models_C FP1(u)$$

17.2. By contradiction, Let us assume
17.2.1. $SN' \models_C FP1(tge)$ [By $\models_C$]

17.2.2. $SN', tge \vDash \forall o.\forall t.\forall \eta.(\neg K_{tge} tagRequest(t, tge, o, \eta) \implies \neg S_{Ag} tag(tge, t, o, \eta))$ [By Implication equivalence]
17.2.3. $SN' \vDash \forall o.\forall t.\forall \eta.(K_{tge} tagRequest(t, tge, o, \eta) \vee \neg S_{Ag} tag(tge, t, o, \eta))$ [By $\neg\neg_i$]
17.2.4. $SN' \vDash \forall o.\forall t.\forall \eta.\neg\neg(K_{tge} tagRequest(t, tge, o, \eta) \vee \neg S_{Ag} tag(tge, t, o, \eta))$ [By Morgan]
17.2.5. $SN' \vDash \forall o.\forall t.\forall \eta.\neg(\neg K_{tge} tagRequest(t, tge, o, \eta) \wedge S_{Ag} tag(tge, t, o, \eta))$

17.3. Let us assume
17.3.1. $SN \vDash_C FP1(tge)$ and $SN \xrightarrow{e} SN'$

17.4. By the Definition of $FR1$, 17.1.2.
17.4.1. $SN' \vDash \neg K_{tge} tagRequest(tgr, tge, o, \eta)$

17.5. By the Definition of $FR1$, 17.1.7.
17.5.1. $SN' \vDash C_{Au} tag(tge, tgr, o, \eta)$[By $\vDash$]
17.5.2. $SN' \vDash E^0_{Au} tag(tge, tgr, o, \eta) \wedge$
$E^1_{Au} tag(tge, tgr, o, \eta) \wedge$
$E^2_{Au} tag(tge, tgr, o, \eta) \wedge$
$E^3_{Au} tag(tge, tgr, o, \eta) \wedge \ldots$ [By $\vDash$]
17.5.3. $\forall j \in Au\ SN' \vDash K_j tag(tge, tgr, o, \eta)$ [Since $o \in Au$ (17.1.5.)]
17.5.4. $SN' \vDash K_o tag(tge, tgr, o, \eta)$

17.6. By 17.4.1., 17.5.4. and 17.2.5. we derive a contradiction.

$\square$

Lemma 3 - Facebook is privacy-preserving

In the proof we consider all possible rules that can be executed and show that none of them will violate FPU, which is the only policy available in the instantiation $\mathcal{FPPF}^{\mathcal{D}}_{\text{FB-Tag}}$.

Proof.

FR1 - Tag

None of the rules can violate FP1 because neither FR1.1 nor FR1.2 increase the audience of any tag. Therefore if FP1 is not in conformance with the current SNM is because of the execution an earlier event. □

FR2 - Accept tag request

FR2.1 would not be executed if $FP1(u) \in \pi_u$ therefore the only case left is FR2.2. □

FR2 - FR2.2.

18. <u>Executing $FR1$ and $FP1$ enabled</u>

18.1. Given
18.1.1. $tge \in Ag$ (owner of the privacy policy $FP1(tge)$)
18.1.2. $picture(o, \eta)$ picture of user $o \in Ag$ and $\eta \in \mathbb{N}$
18.1.3. $Au = audience(picture(o, \eta))$
18.1.4. $e = acceptTagRequest(acptr, tgr, picture(o, \eta))$
18.1.5. We want to prove:

$$SN \vDash_C FP1(u) \text{ and } SN \xrightarrow{e} SN' \text{ then } SN' \vDash_C FP1(u)$$

18.2. By contradiction, let us assume
18.2.1. $SN \vDash_C FP1(u)$ and $SN \xrightarrow{e} SN'$
18.2.2. $SN' \nvDash_C FP1(u)$

18.3. By 18.2.2.
18.3.1. $SN' \nvDash_C FP1(tge)$ [Definition $\vDash_C$]
18.3.2. $SN' \vDash \neg(\forall o.\forall t.\forall \eta.\neg K_{tge} tagRequest(t, tge, o, \eta) \implies \neg S_{Ag} tag(tge, t, o, \eta))$[By Implication equivalence]
18.3.3. $SN' \vDash \exists o.\exists t.\exists \eta.\neg(K_{tge} tagRequest(t, tge, o, \eta) \vee \neg S_{Ag} tag(tge, t, o, \eta))$[By Morgan]
18.3.4. $SN' \vDash \exists o.\exists t.\exists \eta.\neg K_{tge} tagRequest(t, tge, o, \eta) \wedge S_{Ag} tag(tge, t, o, \eta))$[By $\vDash$]
18.3.5. $\exists i \in Au$ s.t. $SN', tge \vDash \exists o.\exists t.\exists \eta.\neg K_{tge} tagRequest(t, tge, o, \eta) \wedge K_i tag(tge, t, o, \eta))$

18.4. By Definition of $FR2.2$, we have that
18.4.1. $SN \vDash K_{acptr} tagRequest(tge, tgr, o, \eta)$[By Definition $FR2.2$, $acptr == tge$]
18.4.2. $SN \vDash K_{tge} tagRequest(tge, tgr, o, \eta)$

18.5. By Definition of $FR2.2$, we have that
18.5.1. $SN' \vDash C_{Au}\, tag(tge, tgr, o, \eta)$ [By $\vDash$]
18.5.2. $SN' \vDash E^0_{Au} tag(tge, tgr, o, \eta) \wedge$
$E^1_{Au} tag(tge, tgr, o, \eta) \wedge$
$E^2_{Au} tag(tge, tgr, o, \eta) \wedge$
$E^3_{Au} tag(tge, tgr, o, \eta) \wedge \ldots$ [By $\vDash$]
18.5.3. $SN' \vDash E^1_{followers(u) \cup \{u\}} tweet(u, \eta)$ [By $\vDash$]
18.5.4. $\forall j \in Au\; SN \vDash K_j\, tag(tge, tgr, o, \eta)$

18.6. By 18.4.2., 18.5.4. and 18.3.5. we derive a contradiction. $\square$

Finally we conclude that $\mathcal{FPPF}^{\mathcal{D}}_{\text{FB-Tag}}$ is a privacy-preserving social network. $\square$

Chapter IV

Model Checking Social Network Models

Raúl Pardo and Gerardo Schneider

Abstract. A *social network service* is a platform to build social relations among people sharing similar interests and activities. The underlying structure of a social networks service is the *social graph*, where nodes represent users and the arcs represent the users' social links and other kind of connections. One important concern in social networks is *privacy*: what others are (not) allowed to *know* about us. The "logic of knowledge" (*epistemic logic*) is thus a good formalism to define, and reason about, privacy policies. In this paper we consider the problem of verifying knowledge properties over *social network models* (SNMs), that is social graphs enriched with *knowledge bases* containing the information that the users know. More concretely, our contributions are: i) We prove that the model checking problem for epistemic properties over SNMs is decidable; ii) We prove that a number of properties of knowledge that are sound w.r.t. Kripke models are also sound w.r.t. SNMs; iii) We give a satisfaction-preserving encoding of SNMs into *canonical* Kripke models, and we also characterise which Kripke models may be translated into SNMs; iv) We show that, for SNMs, the model checking problem is cheaper than the one based on standard Kripke models. Finally, we have developed a proof-of-concept implementation of the model-checking algorithm for SNMs.

IV.1 Introduction

Social networks services (or simply *social networks*) are one of the most popular services on the Internet nowadays. One of the main concerns in social networks is that of privacy: most users are not in full control over what they share, and it is not uncommon that private and personal data is leaked to an unintended audience [51]. These concerns arise because users cannot determine (in a precise manner) who *knows* their personal information. One solution is to provide users with more fine grained control over who knows their information. Epistemic logic or "the logic of knowledge" [29] offers great precision and granularity for modelling and reasoning about the knowledge of the (users or *agents*) in a system.

In [71] we introduced $\mathcal{PPF}$, a formalism based on epistemic logic to specify privacy policies in social networks, and to enable a formal assessment on whether these policies are preserved. $\mathcal{PPF}$ consists of: i) A generic model for social networks (SNMs); ii) A knowledge-based logic ($\mathcal{KBL}$) to reason about the social network and privacy policies; iii) A formal language ($\mathcal{PPL}$) to describe privacy policies (based on $\mathcal{KBL}$). In [67], $\mathcal{PPF}$ was further extended by providing agents with a deductive engine to perform knowledge inferences, and including an operational semantics to model the dynamics of social networks.

$\mathcal{PPF}$ has been specifically designed for privacy policies for real social networks, and that is why the language $\mathcal{PPL}$ and the underlying logic $\mathcal{KBL}$ are interpreted over SNMs and not over Kripke models (*possible-worlds* semantics), which is the "standard" way to give semantics to epistemic logic. In Kripke models the uncertainty of the agents is modelled using an *accessibility relation*. This relation connects all the worlds in the model that an agent considers possible. If a formula is true in all of them, then the agent knows it. This does not correspond to the way users in real world social networks acquire and reason about information. Typically, when a user joins a social network, she knows none or a few facts about it. The system might suggest some friends that are retrieved from the user's phone contacts. As the user makes new friends and they share information, her knowledge starts to grow, and later from this set of accumulated knowledge users may derive new facts.

There are two main advantages in $\mathcal{PPF}$'s design (as opposed to standard Kripke models):

1. **It preserves the original structure of real social networks**. The models in $\mathcal{PPF}$ (SNMs) consist of the *social graph* [27] and a knowledge base per user. The topology of the social graph provides information regarding the relationships between users (e.g., friends, colleagues,...). The knowledge base gives semantics to the modality $K_i\varphi$ (user i knows φ). Knowledge bases are not a new invention, they

are just an instance of the *syntactic* approach to modelling knowledge [40]. This structure is also important from the enforcement point of view since it facilitates the integration of the framework with the target social network.

2. **Checking whether a user knows something must be as efficient as possible**. The privacy policies that users can specify in $\mathcal{PPF}$ talk about knowledge, e.g., "Only my friends can know my location" or "Only my family can know that I am going to my father's birthday party". Therefore, the enforcement of $\mathcal{PPF}$ privacy policies mainly depends on how efficiently these checks are performed. Social networks have millions of users, who disclose tons of information per second. As a consequence, a slow enforcement mechanism would not work in practice. By splitting the users' knowledge in different knowledge bases, the complexity of checking whether a user knows a piece of information can be significantly reduced. In Section IV.6 we study the improvement in complexity of having separated knowledge bases as opposed to standard Kripke semantics.

The properties of knowledge related to human reasoning, present in Kripke models, have been studied for decades and they are well-understood [29]. On the other hand, the properties of knowledge in SNMs have not been throughly studied. Few questions need to be answered: i) What is the relation between SNMs and Kripke models? ii) Does this slightly different representation of knowledge preserve the same properties? iii) Is it possible to determine whether an epistemic formula written in $\mathcal{KBL}$ is satisfied on a given SNM?[1] In this paper we study in depth the answer to these questions providing evidence that $\mathcal{PPF}$ not only offers advantages from the practical point of view, but also models knowledge as traditionally understood and accepted in the epistemic logic literature.

More concretely, our contributions are: i) A proof that model checking $\mathcal{KBL}$ formulae over SNMs is decidable, the algorithm being an implementation of the satisfaction relation for $\mathcal{KBL}$ (Section IV.3); ii) A logical characterisation of a number of properties of knowledge for SNMs including *common* and *distributed knowledge* (Section IV.4). iii) A translation from SNMs into *canonical* Kripke models, together with a proof that satisfaction is preserved (Section IV.5); we also show that it is always possible to reconstruct the original SNM from the canonical Kripke model, by considering the state associated with the characteristic formulae (Section IV.5.1); iv) A formal comparison of the complexity of the model checking problem for SNMs and for Kripke models where we show that the former is more efficient (Section IV.6). Additionally, we provide a proof-of-concept implementation of the model-checking algorithm.[2]

[1] Answering this question will also solve the model checking problem for privacy policies written in $\mathcal{PPL}$, as checking conformance of $\mathcal{PPL}$ is reduced to checking satisfaction of a $\mathcal{KBL}$ formula.

[2] `https://github.com/raulpardo/kbl-model-checker`

Appendices are for reviewing purpose only and should not be considered as part of the paper (it contains the proofs for all theorems and lemmas).

IV.2 Preliminaries

Here we briefly recall First-Order Epistemic Logic [29], social network models and the logic $\mathcal{KBL}$ [67].

IV.2.1 First-Order Epistemic Logic

We start with a set $\mathcal{T}$, consisting of *relation symbols* (p), *function symbols* (f) and *constants symbols* (c). Hereafter we will refer to $\mathcal{T}$ as the *vocabulary.* Each relation and function symbol has an implicit *arity* which corresponds to the number of arguments it takes. Function and relation symbols are interpreted over elements of a domain. We assume an infinite supply of variables, which we write as x, y and so on. We can form terms using constants, variables, and function symbols. Formally, a *term* t is recursively defined as follows: $t ::= c \mid x \mid f(\overrightarrow{t})$, where $\overrightarrow{t}$ represents a list of terms $t_1, \ldots, t_k$. An *atomic formula* is of the form $p(\overrightarrow{t})$ where p is a relation symbol. Let Ag be a set of *agents*, $i \in Ag$ and $G \subseteq Ag$, the syntax of *First-Order Epistemic Logic* (FOEL), denoted as $\mathcal{L}$, is recursively defined as follows [29]:

$$\varphi ::= p(\overrightarrow{t}) \mid \varphi \wedge \varphi \mid \neg\varphi \mid \forall x.\varphi \mid K_i\varphi$$

The remaining epistemic modalities are defined as $S_G\varphi \triangleq \bigvee_{i\in G} K_i\varphi$ and $E_G\varphi \triangleq \bigwedge_{i\in G} \varphi$. The intuitive meaning of the modalities is the following: $K_i\varphi$, agent i knows φ; $E_G\varphi$, everyone in the group G knows φ; $S_G\varphi$, someone in the group G knows φ. The semantics of FOEL formulae is given using *relational Kripke models.* In what follows we omit relational and write Kripke models.

Definition 1 ([29])**.** *A* relational Kripke Model *is a tuple of the form* $\langle S, \pi, \{\mathcal{K}_i\}_{i\in Ag}\rangle$*, where:*

- S *is a non-empty set of* states *(or* worlds*).*
- $\pi : S \to \mathcal{A}$ *is a function that associates to each world a* relation structure *for a fixed vocabulary* $\mathcal{T}$*. As usual,* $\mathcal{A}$ *consists of a domain* D_o*, an assignment of a k-ary relation* $P^{\mathcal{A}} \subseteq D_o^k$ *for each relation symbol, an assignment of a k-ary function* $f^{\mathcal{A}} : D_o^k \to D_o$ *for each function symbol and an assignment of a member* $c^{\mathcal{A}}$ *of the domain for each constant symbol.*
- $\{\mathcal{K}_i\}_{i\in Ag}$ *where* $\mathcal{K}_i \subseteq S \times S$ *is an* accessibility relation *between states.*

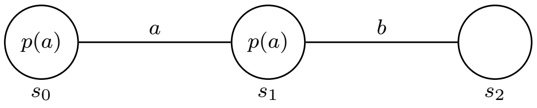


Figure IV.1: Relational Kripke structure

$(M, s) \vDash p(t_1, \ldots, t_k)$	iff	$(t_1, \ldots, t_k) \in P^{\pi(s)}$
$(M, s) \vDash \neg\varphi$	iff	$(M, s) \nvDash \varphi$
$(M, s) \vDash \varphi_1 \wedge \varphi_2$	iff	$(M, s) \vDash \varphi_1$ and $(M, s) \vDash \varphi_2$
$(M, s) \vDash \forall x.\varphi$	iff	for all $v \in dom(\pi(s))$, $(M, s) \vDash \varphi[v/x]$
$(M, s) \vDash K_i\varphi$	iff	$(M, t) \vDash \varphi$ for all t such that $(s, t) \in \mathcal{K}_i$

Table IV.1: Satisfaction relation over Kripke models

Example 1. *Let us consider a Kripke structure consisting of agents a and b, states s_0, s_1 and s_2, a predicate p with arity 1 and relations $\mathcal{K}_a = \{(s_0, s_1), (s_1, s_0)\}$ and $\mathcal{K}_b = \{(s_1, s_2), (s_2, s_1)\}$. We assume here that all relational structures $\pi(s_n)$ have a common domain $D_o = \{a, b\}$, i.e., Ag. Moreover, $a \in P^{\pi(s_0)}$ and $a \in P^{\pi(s_1)}$. Fig. IV.1 shows a graphical representation of the described model.* □

Usually free variables and terms are interpreted using a *valuation* function, which is parametrised with a relational structure depending of the state of the Kripke model in which the formula is evaluated. For simplicity, in this paper we will assume that formulae in $\mathcal{L}$ do not contain free variables (i.e., all variables are quantified) and the interpretation of functions and constants is the same independently of the state where they are evaluated. Thus, we assume that terms are implicitly interpreted and we do not include the valuation function as a parameter in the satisfaction relation below.

Definition 2 ([29])**.** *Given a non-empty set of agents Ag, a relational Kripke model M, a state $s \in M$, agents $i, j, u \in Ag$ and a finite set of agents $G \subseteq Ag$, we define what it means for $\varphi \in \mathcal{L}$ to be* satisfied *by (M, s), written $(M, s) \vDash \varphi$, as shown in Table IV.1.*

We say that a formula φ is *valid in a Kripke model M*, and we write $M \vDash \varphi$, if $\forall s \in M$ $(M, s) \vDash \varphi$. Moreover, we say that φ is *valid*, denoted as $\vDash \varphi$, if for all Kripke models M it holds $M \vDash \varphi$.

Example 2. *Let M be the model presented in Fig. IV.1. It holds that $(M, s_0) \vDash K_a p(a)$, since $p(a)$ holds in s_0 and in all the states accessible for a from s_0 (only s_1). It also holds that $(M, s_1) \vDash \neg K_b p(a)$, since in one of the states that b considers possible $p(a)$ is not true. In particular, $(M, s_2) \vDash \neg p(a)$.*

IV.2.2 $\mathcal{KBL}$ and Social Network Models

$\mathcal{KBL}$ is a *knowledge-based logic* for social networks. It contains all the knowledge modalities presented in $\mathcal{L}$, and additionally, it includes two special types of predicates. The connection and action predicates. Connection predicates represent the "social" connections between users. For instance, friends, colleagues, family, co-workers, and so forth. Action predicates model the permitted actions a user may execute. For example, Alice can send a friend request to Bob or Alice can join events created by Bob. Note that action predicates are not deontic modalities. Hereafter we use $\mathcal{C}$ and Σ to denote sets of indexes for connections and permissions, respectively. As before the set Ag represents a set of agents in the system.

Definition 3. *Given $i, j \in Ag$, a set of predicate symbols $\mathcal{P}$ such that $a_n(i,j)$, $c_m(i,j)$, $p(\overrightarrow{t}) \in \mathcal{P}$ where $m \in \mathcal{C}$ and $n \in \Sigma$, and $G \subseteq Ag$, the syntax of the* knowledge-based logic *$\mathcal{KBL}$ is inductively defined as:*

$$\varphi \quad ::= \quad c_m(i,j) \mid a_n(i,j) \mid p(\overrightarrow{t}) \mid \varphi \wedge \varphi \mid \neg\varphi \mid \forall x.\varphi \mid K_i\varphi$$

As before, the remaining epistemic modalities are defined as $S_G\varphi \triangleq \bigvee_{i\in G} K_i\varphi$ and $E_G\varphi \triangleq \bigwedge_{i\in G}\varphi$.

Terms and atomic formulae are defined as for $\mathcal{L}$. $\mathcal{F}_{\mathcal{KBL}}$ denotes the set of *well-formed formulae* of $\mathcal{KBL}$ (category φ of Def. 3).

Social networks are usually modelled as graphs where nodes represent the users (or agents), and edges represent different relationships among agents or any other social network specific information [27]. These graphs are known as *social graphs*. Here we enrich social graphs with information about the agents knowledge, permissions, connections and privacy policies as defined below.

Definition 4 (Social Network Model)**.** *Given a set of $\mathcal{KBL}$ formulae $\mathcal{F}$, a set of privacy policies Π, and a finite set of agents $Ag \subseteq \mathcal{AU}$ from a universe $\mathcal{AU}$, a* social network model (SNM) *is a social graph of the form $\langle Ag, \mathcal{A}, KB, \pi\rangle$, where*

- *Ag is a nonempty finite set of* nodes *representing the agents of the social network.*
- *$\mathcal{A}$ is a first-order relational structure for the fixed vocabulary of the* SNM*, which as before, consists of a finite domain D_o[3], an assignment of a k-ary relation $P^{\mathcal{A}} \subseteq D_o^{\mathcal{A}}$ for each predicate symbol, an assignment of a k-ary $f^{\mathcal{A}} : D_o^k \to D_o$*

[3]For the sake of clarity in definitions and proofs and w.l.o.g. we have only considered a single finite domain in the formal definition. However, in the rest of the paper we will assume that we have a finite set of finite domains. For instance, we can have D_o consisting of the domain of agents, indexes for posts, indexes for pictures, etc. All the results also hold in SNMs consisting of multiple domains as we consider a finite number of finite domains.

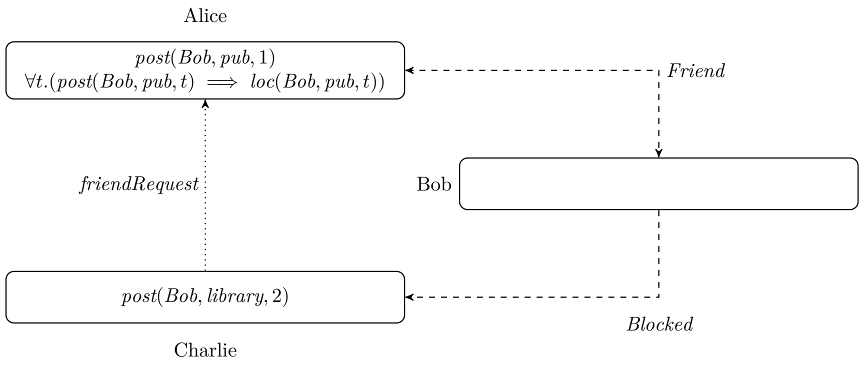


Figure IV.2: Example of Social Network Model

for each function symbol and assignment of a member $c^{\mathcal{A}}$ of the domain for each constant symbol.

- *$KB : Ag \to 2^{\mathcal{F}}$ is a function that returns a finite set of accumulated knowledge for each agent, stored in what we call the* knowledge base *of the agent. We write KB_i to denote $KB(i)$.*
- *$\pi : Ag \to 2^{\Pi}$ is a function that returns a finite set of privacy policies for each agent. We write π_i to denote $\pi(i)$.*

The shape of the relational structure $\mathcal{A}$ depends on the concrete the social network. Connections and permission actions between agents, i.e., edges of the social graph, are represented as families of binary relations, $\{C_i\}_{i \in \mathcal{C}} \subseteq 2^{Ag \times Ag}$ and $\{A_i\}_{i \in \Sigma} \subseteq 2^{Ag \times Ag}$ over the domain of agents. Sometimes, we write an atomic formula, e.g. $friends(a, b)$ to denote that the elements $a, b \in Ag$ belong to a binary relation, *friends*, defined over pairs of agents as expected. $\mathcal{SN}$ denotes the universe of all possible SNMs.

The knowledge base KB_i of each agent i contains the explicit knowledge that the agent has. Besides this explicit knowledge, agents also know anything that can be derived from formulae in their knowledge bases (using the **KD4** axiomatisation of epistemic logic [29]).

Definition 5. *A* derivation *of a formula $\varphi \in \mathcal{F}_{\mathcal{KBL}}$, is a finite sequence of formulae $\varphi_1, \ldots, \varphi_n = \varphi$ where each φ_i, for $1 \leq i \leq n$, is either an instance of the axioms or the conclusion of one of the derivation rules of the* ***KD4*** *axiomatisation which premises have already been derived, i.e., it appears as φ_j with $j < i$.*

Given a set of formulae $\Gamma \in 2^{\mathcal{F}_{\mathcal{KBL}}}$, we write $\Gamma \vdash \varphi$ to denote that φ can be derived from Γ.

Additionally, we impose two assumptions in users' knowledge bases:

i) φ and $\neg\varphi$ cannot be derivable in the same KB_i. It prevents users from having inconsistent knowledge.

ii) If φ is in i's knowledge base, $K_i\varphi$ is also there. In this way we make users aware of their knowledge.

These assumptions are formalised as the following properties:

Definition 6 (Knowledge Consistency). *For all $i \in Ag$ and formulae $\varphi \in \mathcal{F}_{\mathcal{KBL}}$, if $KB_i \vdash \varphi$ then $KB_i \nvdash \neg\varphi$.*

Enforcing knowledge consistency is straightforward. Before adding any formula φ to KB_i we check that $KB_i \cup \{\varphi\} \nvdash \neg\varphi$.

Definition 7 (Self-Awareness). *For all $i \in Ag$ and formulae $\varphi \in \mathcal{F}_{\mathcal{KBL}}$, if $KB_i \vdash \varphi$ then $KB_i \vdash K_i\varphi$.*

Remark 1. Self-awareness is not equivalent to the necessitation rule in **KD4**. Necessitation states that if a φ is provable from no assumptions then $K_i\varphi$ is provable from no assumptions as well [29]. That is, $\dfrac{\vDash \varphi}{\vDash K_i\varphi}$. It requires φ to be a tautology. On the other hand, self-awareness states that if φ is derivable from i's knowledge, then $K_i\varphi$ is also derivable. For example, $\varphi \vee \neg\varphi$ is provable from no assumptions. Therefore, from axiom A1 it is derivable $KB_i \vdash \varphi \vee \neg\varphi$ for all KB_i. Consequently, by necessitation it also holds that $KB_i \vdash K_j\varphi \vee \neg\varphi$ for all KB_i and $j \in Ag$. However, consider now a predicate $p(\overrightarrow{t})$ which is not derivable from no assumptions. It does not hold that $KB_i \vdash p(\overrightarrow{t})$ for all KB_i. There is no axiom which includes $p(\overrightarrow{t})$ in the set of derivations of $\vdash$. Nevertheless, self-awareness says that if $KB_i \vdash p(\overrightarrow{t})$ then $KB_i \vdash K_ip(\overrightarrow{t})$. Note that, unlikely necessitation, we use the same agent i in KB_i and $K_ip(\overrightarrow{t})$.

Example 3. *Let SN be an SNM consisting of three agents Alice, Bob and Charlie, $Ag = \{Alice, Bob, Charlie\}$; the friend request action, $\Sigma = \{friendRequest\}$; and the connections Friend and Blocked, $\mathcal{C} = \{Friend, Blocked\}$. Here, we define D_o to be a finite set of timestamps.*

Fig. VI.1 shows a graphical representation of SN. In this model the dashed arrows represent connections. Note that the Friend connection is bidirectional, i.e., Alice is friend with Bob and vice versa. On the other hand, it is also possible to represent unidirectional connections, as Blocked; in SN Bob has blocked Charlie. Permissions are represented using a dotted arrow. In this example, Charlie is able to send a friend request to Alice.

The predicates inside each node represent the agents' knowledge, e.g., Alice has $post(Bob, pub, 1)$ in her knowledge base, meaning that she knows that Bob posted at

$SN \models p(\overrightarrow{t})$	iff	$p(\overrightarrow{t}) \in KB_e$
$SN \models c_m(i,j)$	iff	$(i,j) \in C_m$
$SN \models a_n(i,j)$	iff	$(i,j) \in A_n$
$SN \models \neg\varphi$	iff	$SN \not\models \varphi$
$SN \models \varphi \wedge \psi$	iff	$SN \models \varphi$ and $SN \models \psi$
$SN \models \forall x.\varphi$	iff	for all $v \in D_o$, $SN \models \varphi[v/x]$
$SN \models K_i\varphi$	iff	$KB_i \vdash \varphi$

Table IV.2: $\mathcal{KBL}$ satisfaction relation

time 1 that he was in a pub. Similarly, Charlie's knowledge base contains the predicate $post(Bob, library, 2)$ meaning that at time 2 Bob posted that he was in the library. Agents' nodes can also contain more complex $\mathcal{KBL}$ formulae that may increase their knowledge. For instance, Alice knows $loc(Bob, pub, 1)$ implicitly. Alice can in fact derive it by Modus Ponens*, from $post(Alice, pub, 1)$ and $\forall t.(post(Alice, pub, t) \implies loc(Bob, pub, t))$. The variable t ranges over D_o, which, as mentioned earlier, consists in a finite set of timestamps. Being able to derive $loc(Bob, pub, 1)$ means that Alice knows that Bob's location at time 1 was a pub.*

The satisfaction relation for $\mathcal{KBL}$ formulae, interpreted over SNMs, is defined as follows.

Definition 8. *Given an SNM $SN = \langle Ag, \mathcal{A}, KB, \pi\rangle$, agents $i, j \in Ag$, formulae $\varphi, \psi \in \mathcal{F}_{\mathcal{KBL}}$, a finite set of agents $G \subseteq Ag$, $m \in \mathcal{C}$ and $n \in \Sigma$, the* satisfaction relation *$\models \subseteq \mathcal{SN} \times \mathcal{KBL}$ is defined in Table IV.2.*

The intuition behind the semantic definition of the knowledge modality is different in $\mathcal{KBL}$ from that of epistemic logic. As shown in Table IV.1, the accessibility relation in Kripke models captures the *uncertainty* of the agents. It models all the states that an agent consider possible and knowledge is acquired when a given formula is true in all those states. In SNMs, knowledge is explicitly present in the knowledge bases of the agents, hence modelling what the agents know rather than what they consider possible. A given formula is known by an agent if it is present in her knowledge base or if she can derive it from her knowledge. We use a special agent called *environment* (or simply e) which defines the truth of atomic formulae of the type $p(\overrightarrow{t})$. The environment's knowledge base (KB_e) contains all predicates which are true in the real world. For instance, $location(Alice, Sweden)$ is in KB_e only if Alice's location is Sweden or, similarly, only if Bob's age is 20 the predicate $age(Bob, 20)$ is in KB_e.

Example 4. *Let SN be the SNM in Fig. VI.1. As described in Example 1, Alice knows Bob posted that at time 1 he was in a pub, meaning that $SN \models K_{Alice}post(Bob, pub, 1)$*

holds. Indeed, it holds since $post(Bob, pub, 1)$ *is in the knowledge base of Alice, i.e.,* $post(Bob, pub, 1) \in KB_{Alice}$ *and therefore it can be derived* $KB_{Alice} \vdash post(Bob, pub, 1)$ *(1). Though not explicitly stated, it is possible for Alice to derive that Bob's location at time 1 was a pub, meaning that* $SN \vDash K_{Alice} loc(Bob, pub, 1)$ *(2) should hold. Following the semantics of* K_i *in Table IV.2, the previous formula is true iff* $KB_{Alice} \vdash loc(Bob, pub, 1)$*. Fig. VI.1 shows that* KB_{Alice} *contains the formula* $\forall t.(post(Bob, pub, t) \implies loc(Bob, pub, t))$ *(3)—where* t *is a timestamp —therefore the deductive engine derives* $post(Bob, pub, 1) \implies loc(Bob, pub, 1)$ *(4). From (1) and (4), by* modus ponens *we can derive* $loc(Bob, pub, 1)$*, i.e.,* $KB_{Alice} \vdash loc(Bob, pub, 1)$*, hence (2) holds.*

IV.3 Model checking SNMs

In this section we present a model checking algorithm that directly implements the semantics of $\mathcal{KBL}$ in Table IV.2, and we show that model checking is decidable under the following assumptions:

Assumption 1. *All domains are finite.*

Assumption 2. *All functions are computable.*

These assumptions are present in all real social networks. Domains in SNMs might be, the set of users, posts, pictures, likes, tags and so on. In practice at any moment in time there is a finite amount of any of these elements. Consequently, when having a universal quantification over a domain it is reasonable to consider only the finite set of elements in the domain at that concrete moment in time. Furthermore, we assume that functions in $\mathcal{KBL}$ terms must be computable. As mentioned in the introduction, $\mathcal{KBL}$ is a logic embedded in a framework to express privacy policies. The framework includes the notion of instantiation where all the elements of SNMs are instantiated for a concrete social network. For instance, in [71] we presented the instantiations of Facebook and Twitter. In these instantiations functions were used to retreive information, e.g., $followers(u)$ which returns all the followers of the user or $friends(u)$ which returns all the friends of u. Another type of functions could be $weather(London)$ or $location(u)$, which return the current weather in London and u's current location, respectively. Therefore, computable functions are enough for the practical use of the logic.

Theorem 1. *Let* SN *be an SNM and* $\varphi \in \mathcal{F}_{\mathcal{KBL}}$ *be a formula. Determining whether* $SN \vDash \varphi$ *is decidable.*

Proof. We show decidability of the model checking problem for $\mathcal{KBL}$ by presenting an algorithm which implements the semantics of Table IV.2,

First, we expand the universal quantifiers in φ by inductively transforming each subformula $\forall x.\varphi'$ into a conjunction with one conjunct $\varphi'[v/x]$ for each element v of the domain D_o. Given that the domain is finite (see Assumption 1), it always terminates and results in a quantifier free formula. Secondly, we compute all functions and replace all constants with an element of the domain according to the assignment in $\mathcal{A}$. From Assumption 2, we can deduce that this step always terminates. After this step we are left with a quantifier free formula without functions or constant symbols. Finally, we inductively show that all the elements of the formula (see Def. 3) can be computed.

- Checking $c_m(i,j)$ and $a_n(i,j)$ can be performed in constant time, simply by checking $(i,j) \in C_m$ or $(i,j) \in A_n$, respectively.
- Checking $p(\overrightarrow{t})$ requires the query $p(\overrightarrow{t}) \in KB_e$ to the environment's knowledge base. It can be performed in constant time.
- $\neg\varphi$ and $\varphi_1 \wedge \varphi_2$ can be done in constant time, using the induction hypothesis.
- $K_i\varphi$ requires a query to the epistemic engine to determine $KB_i \vdash \varphi$. Solving the previous query is a decidable problem [29].

The algorithm goes recursively from the top most element of φ to the bottom. $\square$

In Section IV.6 we study the complexity of this algorithm and compare it to that of model checking in traditional Kripke models. Nevertheless, in order to provide a fair comparison, we first show that the same set of properties of knowledge that are sound w.r.t. Kripke models are also sound w.r.t. SNMs.

IV.4 Properties of Knowledge in SNMs

Here we explore properties of knowledge in SNMs. In particular, we consider the axioms of some of the standard axiomatisations for epistemic logic, and prove that such axioms are sound with respect to SNMs.

In [29] Fagin *et al.* show which properties of knowledge are sound w.r.t. Kripke models depending on the type of accessibility relation of the model. For instance, the following axiom is sound w.r.t. the set of Kripke models where the accessibility relation is reflexive: (A3) $K_i\varphi \implies \varphi$.

These properties of knowledge comprise the different axiomatisations of epistemic logic. In SNMs the properties of knowledge will depend on the axiomatisation from epistemic logic [29] that we choose for $\vdash$. As we described in Def. 6, $\vdash$ includes all the axioms and derivation rules from **KD4**.

In epistemic logic one can talk about *knowledge* or *belief* depending on the properties (or axiomatisations) that are sound w.r.t. a particular set of Kripke models. Axiom A3 is commonly called *Knowledge axiom*. It means that the facts agents know are true. When this axiom is not present, the "knowledge" of the agents is regarded as belief. As you might have noticed, in SNMs the truth of the facts that the agents know is not linked to whether they are true or not. For example, imagine that Alice knows that Bob and Charlie are friends, i.e., $K_{Alice} friend(Bob, Charlie)$, which is true iff $KB_{Alice} \vdash friend(Bob, Charlie)$. This is not connected to the actual truth of the predicate $friend(Bob, Charlie)$, which holds iff $(Bob, Charlie) \in C_{Friend}$. When the knowledge axiom is not present, some philosophers argue that it is required that the beliefs of the agents are consistent. This is captured by the following axiom, where $\perp$ represents *falsum*: (D) $\neg K_i \perp$.

In Kripke models, axiom D is present when the accessibility relation is serial [29]. In SNMs, we assume agents' knowledge bases to be consistent (see Def. 6). Therefore, $\perp$ cannot be derived.

Lemma 1. *Axiom D is sound with respect to SNMs.*

As we mentioned in the introduction, $\mathcal{KBL}$ and SNMs were developed in the context of a privacy policy framework for social networks [71, 67]. In privacy policies it is more natural to write "Alice cannot know my location" than "Alice cannot belief my location". Because of this, we chose to talk about knowledge, even though we are dealing with an axiomatisation for belief.

The most basic set of properties for Kripke models, i.e., the set of properties that are sound w.r.t. Kripke models with no conditions in their accessibility relation, is the **K** axiomatisation [29]. It consists of two axioms and two inference rules. Given $\varphi \in \mathcal{L}$ and $i \in Ag$,

A1. All (instances of) first-order tautologies,
A2. $(K_i\varphi \wedge K_i(\varphi \implies \psi)) \implies K_i\psi$,
R1. From φ and $\varphi \implies \psi$ infer ψ,
R2. From φ infer $K_i\varphi$ where φ must be provable from no assumptions.

Lemma 2. ***K*** *is sound with respect to SNMs.*

The axioms and inferences rules of **K**, together with axiom D comprises the axiom system **KD**. Nevertheless, there exist two more axioms that are normally present in knowledge and belief axiomatisations, the so called *positive introspection* (A4) and *negative introspection* (A5) [29]. The former expresses that agents in the system are aware of their knowledge, the latter means that agents know everything that they do not know. Given $\varphi \in \mathcal{L}$ and $i \in Ag$

A4. $K_i\varphi \implies K_iK_i\varphi$,
A5. $\neg K_i\varphi \implies K_i\neg K_i\varphi$.

Lemma 3. *Axiom A4 is sound with respect to SNMs.*

Lemma 4. *Axiom A5 is not sound with respect to SNMs.*

A4 follows from our assumption that agents are self-aware of their knowledge (see Def.7). On the other hand, A5 does not follow given the current set of assumptions in knowledge bases. An agent's knowledge base does not contain any knowledge regarding what she does not know, unless it is explicitly inserted.

The axiomatisation **K** together with axioms D and A4 forms the so-called **KD4** axiomatisation. We thus have the following result for SNMs.

Theorem 2. ***KD4*** *is sound with respect to SNMs.*

Common Knowledge

Here we introduce the notion of *common knowledge*, which we represent using the modality C_G where G is a group of agents. A fact becomes common knowledge when everybody knows it, and also, everyone knows that everyone knows it, and so forth. This is a useful concept in the social network setting. Consider the effect of publishing a post $p(\overrightarrow{t})$ in a social network. After posting, the owner of the post and the audience will know the post, $E_{\{owner\}\cup audience}\ p(\overrightarrow{t})$. Moreover, the owner also will know that everyone who was included in the audience will know the post, $K_{owner}E_{audience}\ p(\overrightarrow{t})$. But even more, each of the users in the audience will know that each other knows the post, i.e. $E_{\{owner\}\cup audience}E_{\{owner\}\cup audience}\ p(\overrightarrow{t})$ and so on. The traditional definition of common knowledge [29] over Kripke models accurately captures the described effect. Given a Kripke model M, a state $s \in M$, a formula $\varphi \in \mathcal{L}$ and a set of agents G, common knowledge is defined as follows:

$$(M,s) \models C_G\varphi \text{ iff } (M,s) \models E^k_G\varphi \text{ for } k = 1\ldots$$

where $E^0_G\varphi = \varphi$ and $E^{k+1}_G\varphi = E_G\varphi E^k_G\varphi$. The definition of common knowledge for SNMs is analogous to the one above.

Definition 9. *Given an SNM SN, a formula $\varphi \in \mathcal{F}_{\mathcal{KBL}}$ and a set of agents G, common knowledge is defined as follows:*

$$SN \models C_G\varphi \textit{ iff } SN \models E^k_G\varphi \textit{ for } k = 1\ldots$$

Given formulae $\varphi, \psi \in \mathcal{L}$, the set $G \subseteq Ag$ and $i \in Ag$, the following axiomatisation characterises common knowledge [29]:

C1. $E_G\varphi \iff \bigwedge_{i\in G} K_i\varphi$,

C2. $C_G\varphi \iff E_G(\varphi \wedge C_G\varphi)$,

RC1. From $\varphi \implies E_G(\psi \wedge \varphi)$ infer $\varphi \implies C_G\psi$ where $\varphi \implies E_G(\psi \wedge \varphi)$ must be provable from no assumptions.

Lemma 5. *The axioms C1 and C2, and the rule RC1 are sound w.r.t. SNMs.*

Distributed Knowledge

In this section we introduce the distributed knowledge operator, represented by the modality D_G. A fact becomes distributed knowledge in the group of agents G when it is known by combining the knowledge of all individual agents. It can be seen as a wise agent. In Kripke models, distributed knowledge is defined by removing possible states, i.e., removing uncertainty. Formally,

$$(M, s) \models D_G\varphi \text{ iff } (M, t) \models \varphi \text{ for all } t \text{ such that } (s, t) \in \bigcap_{i\in G} \mathcal{K}_i.$$

We define distributed knowledge as the union of all the explicit knowledge that all the agents in G have and everything that can be derived from it.

Definition 10 (Distributed knowledge)**.** *Given an SNM SN, a formula $\varphi \in \mathcal{F}_{\mathcal{KBL}}$ and a set of agents G, distributed knowledge is defined as follows:*

$$SN \models D_G\varphi \text{ iff } \bigcup_{i\in G} KB_i \vdash \varphi.$$

The following axioms characterise distributed knowledge [29]:

D1. $D_{\{i\}}\varphi \iff K_i\varphi,\ i = 1, \ldots, n$,

D2. $D_G\varphi \implies D_{G'}(\varphi)$ if $G \subseteq G'$,

DA2 and DA4. Axioms A2 and A4 of **KD4**, K_i with D_G in each axiom.

Note that axiom D is not required because we work with a belief axiomatisation [29]. Therefore, it is possible for a group of agents to have inconsistent distributed beliefs. In what follows, we show that this axiomatisation for Kripke models is sound with respect to SNMs as well.

Lemma 6. *Axioms D1 and D2, together with the axioms A2 and A4 of the* ***KD4****-axiomatisation (replacing the modality K_i with the modality D_G) are sound w.r.t. SNMs.*

IV.5 Translation of SNMs into Kripke Models

In this section, we show that SNMs can be encoded into Kripke models. Our proof is constructive, starting from an SNM we give a procedure to build a *canonical* Kripke model, and we prove that satisfaction is preserved when interpreting $\mathcal{KBL}$ formulae as epistemic logic formulae.

For epistemic logic, Fagin *et al.* show that it is possible to construct a canonical Kripke model which satisfies a given formula φ [29], provided that φ is consistent with respect to some of the axiomatisations of knowledge. A formula φ is ***KD4**-consistent* if $\neg\varphi$ cannot be derived. A set of formulae is **KD4**-consistent if the conjunction of all the formulae in the set is **KD4**-consistent. We say that a set of formulae Φ is *maximal **KD4**-consistent* with respect to the language $\mathcal{L}$, if Φ is **KD4**-consistent and for all φ in $\mathcal{L}$ but not in Φ, the set $\Phi \cup \{\varphi\}$ is not **KD4**-consistent. In what follows, we describe the procedure of how to construct a canonical Kripke model for a **KD4**-consistent formula. We will follow a similar approach when translating SNMs into Kripke models.

Definition 11 (Canonical Kripke model for **KD4**[29])**.** *Consider a* **KD4**-consistent *formula φ. Let $Sub(\varphi)$ be the set of all subformulae of φ. We define $Sub^+(\varphi)$ to be the set of all subformulae and their negations, i.e. $Sub^+(\varphi) = Sub(\varphi) \cup \{\neg\psi \mid \psi \in Sub(\varphi)\}$. We also define $Con(\varphi)$ to be the set of maximal **KD4**-consistent subsets of $Sub^+(\varphi)$. Given a set of formulae $\Theta \subseteq \mathcal{L}$, we define $\Theta/K_i = \{\varphi \mid K_i\varphi \in \Theta\}$. The canonical Kripke model for φ is defined as follows: $M_\varphi = \langle S_\varphi, \pi, \{\mathcal{K}_i\}_{i \in Ag}\rangle$ where $S_\varphi = \{s_\Theta \mid \Theta \in Con(\varphi)\}$, $\mathcal{K}_i = \{(s_\Theta, s_\Psi) \mid \Theta/K_i \subseteq \Psi/K_i,\ \Theta/K_i \subseteq \Psi\}$ and*

$$\pi(s_\Theta)(p(t_1,\dots,t_k)) = \begin{cases} \textbf{\textit{true}} & \textit{if}\ \ p(t_1,\dots,t_k) \in \Theta \\ \textbf{\textit{false}} & \textit{if}\ \ p(t_1,\dots,t_k) \notin \Theta \end{cases}$$

Fagin *et al.* show that φ is satisfiable in the resulting canonical Kripke model [29, Theorem 3.2.4]. The set of Kripke models that are sound and complete with respect to **KD4** are the ones with a serial and transitive accessibility relation. The accessibility relation of the previous canonical Kripke model is, as shown in [29, Theorem 3.2.4], serial and transitive. We denote the set of Kripke models with the previous type of accessibility relation as $\mathcal{M}^{lt}$.

The canonical Kripke model will have at most $2^{|\varphi|}$ states, as shown in [29, Theorem 3.2.4] where $|\varphi|$ is the length of the formula φ. Even though it is finite, this approach of constructing a Kripke model can lead to an exponential growth of the size of the model. For example, if we assume that the knowledge of the agents increases monotonically, i.e., agents do not forget any knowledge they have previously obtained, then the size of φ will have a lower bound, from which its size will only grow, and consequently, the size of the corresponding canonical Kripke model. In what follows, we define a function which takes an SNM and converts it into the corresponding canonical Kripke model.

First we describe how to construct a set containing all the true formulae in an SNM, called the characteristic set of the social network.

Definition 12. *The* characteristic set *of an SNM SN, denoted as Φ_{SN}, is constructed as follows:* $\Phi_{SN} = \{p(\vec{t}) \mid p(\vec{t}) \in KB_e\} \cup \{K_i\varphi \mid \varphi \in KB_i\} \cup \{c(i,j) \mid (i,j) \in C_c, c \in \mathcal{C}\} \cup \{a(i,j) \mid (i,j) \in A_a, a \in \Sigma\}$.

Moreover, we define the *characteristic formula* of an SNM.

Definition 13. *Given a characteristic set, Φ_{SN}, of an SNM SN, its* characteristic formula*, denoted as φ_{SN}, is defined as $\varphi_{SN} = \bigwedge_{\psi \in \Phi_{SN}} \psi$.*

We will use the characteristic formula of an SNM to create the corresponding Kripke model, therefore we must show that this formula is **KD4**-consistent.

Lemma 7. *For all $SN \in \mathcal{SN}$, φ_{SN} is **KD4**-consistent.*

We are now ready to provide our translation from SNMs into canonical Kripke models.

Definition 14 (Kripke transformation function)**.** *Let $\mathcal{KT} : \mathcal{SN} \to \mathcal{M}^{lt}$ be a function which takes an SNM and converts it to the corresponding Kripke model as follows. Given an $SN \in \mathcal{SN}$, $\mathcal{KT}(SN)$ is defined as follows: 1) Construct Φ_{SN} as defined in Def. 12; 2) Construct φ_{SN} as defined in Def. 13; 3) Return the resulting canonical Kripke model of φ_{SN} as defined in Def. 11.*

We thus have our main theorem.

Theorem 3. *If a formula φ is satisfied in an SNM SN then φ is satisfied in the Kripke model $\mathcal{KT}(SN)$.*

IV.5.1 Translation of Kripke Models into SNMs

Note that, in general, it is not possible to translate arbitrary Kripke models into SNMs. One of the reasons is that in Kripke models there exists only one type of predicate, which is always interpreted in the same way, whereas in SNMs, there are three types of predicates. We cannot even translate back canonical Kripke models constructed using $\mathcal{KT}$. To see why let us consider a canonical Kripke model with the following characteristic set of formulae $\{K_{Alice}loc(Bob, library), friend(Alice, Bob)\}$. We know that the predicate $loc(Bob, library)$ belongs to Alice's knowledge base, since it is under the scope of a knowledge modality. However, we cannot know the type of the predicate *friend*$(Alice, Bob)$, it could be part of a connection relation, action relation or simply be a regular predicate which should appear in the environment's knowledge base.

That said, we show here that it is in fact always possible to reconstruct the original SNM from the canonical Kripke model, if we slightly modify our translation function $\mathcal{KT}$. Let Φ^m_{SN} be a *marked characteristic set*, which is a characteristic set as defined in Def. 12, but having the predicates annotated so that their type can be syntactically identified. For example, if the predicate above *friend*$(Alice, Bob)$ is a connection predicate, it would be converted to *co_friend*$(Alice, Bob)$. We can now define $\mathcal{KT}^m$ to be a Kripke transformation function as in Def. 14, except for the input characteristic set, which is replaced by Φ^m_{SN}. Given that we can uniquely identify the type of the predicates it is trivial to define a function that takes a Kripke model constructed using $\mathcal{KT}^m$ and returns the equivalent SNM. The function proceeds as follows: firstly, it searches for all the agents present in all formulae and subformulae in Φ^m_{SN} and creates one node per agent; secondly, it puts regular predicates in the environment's knowledge base; thirdly it creates relations between agents for each connection and permission predicate; finally, for all formulae of the form $K_i\varphi$ it includes φ in i's knowledge base. See Appendix IV.A.4 for the formal definitions of Φ^m_{SN}, $\mathcal{KT}^m$ and the SNM construction).

We also show that satisfaction is preserved between a marked canonical Kripke model and its original SNM when formulae are evaluated in the state corresponding to the marked characteristic set ($s_{\Phi^m_{SN}}$).

Theorem 4. *If a formula φ is satisfied in the state $s_{\Phi^m_{SN}}$ of a Kripke model $\mathcal{KT}^m(SN)$ then φ is satisfied in the SNM SN.*

IV.6 Model checking complexity

In [29], Fagin *et al.* prove that the complexity of the model checking problem for **KD4** (without common and distributed knowledge) is PSPACE-complete for n agents where $n > 1$ and NP-complete for one agent. They also prove that for a model $M = (S, \pi, \mathcal{K}_1, \ldots, \mathcal{K}_n)$ *"There is an algorithm that, given a structure M, a state s of M and a formula $\varphi \in \mathcal{L}$, determines, in time $O(||M|| \times |\varphi|)$, whether $(M, s) \vDash \varphi$"* (see [29, Proposition 3.2.1]) where $||M||$ is the sum of all the states in S and the number of pairs in all $\mathcal{K}_i$, and $|\varphi|$ is the length of the formula defined as usual. This algorithm is not optimal, but the result is useful to compare the model checking problem in SNMs and the Kripke models constructed using our translation.

Let $M_{\varphi_{SN}}$ be the model $\mathcal{KT}(SN)$ for an SNM *SN*. The complexity of the model checking problem of a formula φ in the previous model is $O(||M_{\varphi_{SN}}|| \times |\varphi|)$. $M_{\varphi_{SN}}$ has size at most $2^{|\varphi_{SN}|}$ (see Section IV.5), therefore it holds $||M_{\varphi_{SN}}|| \leq 2^{|\varphi_{SN}|}$. Thus, for simplicity and w.l.o.g. the above may be rewritten as $O(2^{|\varphi_{SN}|} \times |\varphi|)$.

In what follows we study the complexity of the model checking problem in $\mathcal{KBL}$. The proof of Theorem 1 describes an algorithm to determine whether $SN \vDash \varphi$. We consider $\mathcal{KBL}$ without common and distributed knowledge, since the complexity for Kripke models mentioned at the beginning of the section also excludes these modalities. For simplicity in the complexity analysis and w.l.o.g. we only consider quantifier free formulae which do not contain functions.

Let M_{KB_i} be the canonical Kripke model resulting from the conjunction of all formulae in agent's i knowledge base using our translation, the complexity of the model checking problem is given by the function *checking complexity* (**cc**):

$$\begin{array}{rclrcl}
\mathbf{cc}(p(\overrightarrow{t})) & = & c & \mathbf{cc}(\neg\varphi) & = & 1 + \mathbf{cc}(\varphi) \\
\mathbf{cc}(c(i,j)) & = & c & \mathbf{cc}(\varphi_1 \wedge \varphi_2) & = & 1 + \mathbf{cc}(\varphi_1) + \mathbf{cc}(\varphi_2) \\
\mathbf{cc}(a(i,j)) & = & c & \mathbf{cc}(K_i\varphi) & = & O(||M_{KB_i}|| \times |\varphi|)
\end{array}$$

where c is an upper-bound in the cost of checking satisfaction of predicates in the environment's knowledge base, connection predicates and action predicates. Negation and conjunction need one step plus the complexity of checking satisfaction of their subformulae. Finally, satisfaction of $K_i\varphi$ depends on checking $KB_i \vdash \varphi$, which requires solving the model checking problem as defined for Kripke models. Therefore it has the same complexity. Let $outerK : \mathcal{F}_{\mathcal{KBL}} \to 2^{\mathcal{F}_{\mathcal{KBL}}}$ be a function that takes a $\mathcal{KBL}$ formula and returns the set of subformulae where K_i is the top most operator and it is not under the scope of a knowledge modality. For example, $outerK(K_a(p(s) \wedge K_b q(s)) \wedge p(u) \wedge \neg K_b r(s) \wedge K_c u(v)) = \{K_a(p(s) \wedge K_b q(s)), K_b r(s), K_c u(v)\}$. Note that $K_b q(s)$ is not part of the set because it is under the scope of K_a. The complexity of checking whether a formula φ is satisfiable in an SNM is

$$O(\sum_{K_i\varphi_i \in outerK(\varphi)} (||M_{KB_i}|| \times |\varphi_i|) + m_\varphi)$$

where $m_\varphi \in \mathbb{N}$. The characteristic formula of an agent's knowledge base is the conjunction of all its knowledge, which we denote as φ_{KB_i}. As before, it holds that $||M_{KB_i}|| < |2^{\varphi_{KB_i}}|$, which we use again for the complexity of the problem

$$O(\sum_{K_i\varphi_i \in outerK(\varphi)} (2^{|\varphi_{KB_i}|} \times |\varphi_i|) + m_\varphi).$$

The intuition is as follows: m_φ is the cost of checking predicates, conjunctions and negations in φ, which we assume to be some constant that depends on the length of

φ. Besides, $\sum_{K_i\varphi_i \in outerK(\varphi)} (2^{|\varphi_{KB_i}|} \times |\varphi_i|)$ is the cost of checking each subformula φ_i in the knowledge base of the corresponding agent. In short, we have replaced checking satisfaction of φ in a complete model of the social network to checking satisfaction of subformulae of φ in the corresponding knowledge bases of the agents.

Checking the parts of φ that only contain predicates and logical connectives has very similar complexity in both models. In the canonical Kripke model of an SNM *SN*, the state corresponding to the characteristic set ($s_{\Phi_{SN}}$) contains all true predicates (see Def. 11). Similarly, in SNMs it is only needed to check the environment's knowledge base, and the connection and action relations (see Table IV.2). In both cases the complexity is determined by the length of this particular part of φ. Therefore, in order to compare the complexity of the model checking problem, we only focus on the parts of the formula that are under the scope of a knowledge modality. Given a formula φ, let φ^K be the conjunction of the subformulae starting with a K_i modality (for any $i \in Ag$), formally, $\varphi^K \triangleq \bigwedge_{\psi \in outerK(\varphi)} \psi$. Thus the complexity of the model checking problem in Kripke models is reduced to $O(2^{|\varphi_{SN}|} \times |\varphi^K|)$, and in SNMs it is $O(\sum_{K_i\varphi_i \in outerK(\varphi)}(2^{|\varphi_{KB_i}|} \times |\varphi_i|))$. To formally compare the complexity of the problem in both models we prove the following.

Lemma 8. *Given $SN \in \mathcal{SN}$ and a formula φ the following holds*

$$O(\sum_{K_i\varphi_i \in outerK(\varphi)} (2^{|\varphi_{KB_i}|} \times |\varphi_i|)) < O(2^{|\varphi_{SN}|} \times |\varphi^K|).$$

The previous lemma shows that it is always more efficient to check satisfaction of a formula φ in SNMs. Intuitively, it shows that it is more efficient to construct Kripke models representing the agents' knowledge base and locally check the corresponding subformulae, than constructing the complete Kripke model to check the conjunction of the mentioned subformulae. The difference in complexity becomes more apparent as less agents are involved in the knowledge modalities of φ. When an agent is not mentioned in φ her knowledge base is disregarded. For instance, in the SNM of Fig. VI.1 checking $K_{Charlie} loc(Bob, pub, 1)$ requires (at most) $2^4 + 5 = 21$ steps where 4 is the size of the formula in *Charlie*'s knowledge base and 5 is the size of $K_{Charlie} loc(Bob, pub, 1)$, whereas in the corresponding canonical Kripke model it requires (at most) $2^{4+14+12} + 5 = 1073741829$ steps where 14 is the size of the conjunction of all the formulae in the knowledge base of *Alice* (assuming that the domain of x only has one element), and 12 is the size of the predicates $friend(Alice, Bob)$, $friend(Bob, Alice)$, $blocked(Bob, Charlie)$ and $friendRequest(Charlie, Alice)$.

IV.7 Related work

The use epistemic logic to model knowledge in social networks is not new. One line of work consists in using two dimensional modal logic. It relies on Kripke models where the knowledge of the agents in the social network is encoded using an accessibility relation, and friendship is represented using a symmetric irreflexive relation between agents [85, 84]. Other epistemic logics include a public (and private) announcement operator to study diffusion of information in the network [80, 15, 16]. Permission and knowledge has also been merged in the so called deontic-epistemic logic [3]. For a detailed comparison among these logics and $\mathcal{KBL}$ we refer to the work by Pardo & Schneider [71, 67] and references therein.

There exist several model checkers for epistemic logic that perform efficiently in rather large scenarios [33, 25, 54]. However, as shown in this paper, model checking in the canonical Kripke model constructed from an SNM has higher complexity than in the SNM.

On the other hand, the model checking algorithm presented in this paper requires checking whether $KB_i \vdash \varphi$. As mentioned in Section IV.2.2, this check can be resolved by using any of the existing model checkers or SAT solvers for epistemic logic. For this reason, any improvement in the efficiency of the model checking problem in Kripke models, will also be improve the performance when checking formulae in the individual knowledge bases of each agent. In addition, local checks in different knowledge bases can easily be parallelised. For instance, if there is one process per knowledge base, formulae regarding different agents' knowledge can be checked in parallel in the corresponding knowledge bases. To the best of our knowledge, there are no parallel model checkers for epistemic logic.

IV.8 Final Discussion

We have proved that the model checking problem in SNMs is decidable. We have shown the relation between SNMs and Kripke models. Concretely, we have proven that the belief axiomatisation **KD4**, which was originally defined for epistemic logic and naturally models agents' reasoning, is sound w.r.t. SNMs. We have provided a translation of SNMs models into canonical Kripke models and proved that satisfaction of any formula in the SNM is preserved in the corresponding Kripke model. We have also provided a translation from the canonical Kripke structure (obtained from our translation from SNMs) into the original SNM. We have proven that all formulae are satisfied in the state corresponding to the characteristic set of the SNM in the Kripke model are also satisfied in the original SNM. Finally, we showed the model checking problem in SNMs using our algorithm is more efficient than using the standard Kripke semantics.

We conjecture that arbitrary Kripke models (in the frame of models with serial and transitive relations) can be translated to SNMs. However, to preserve satisfaction the translation would generate several SNMs from a given Kripke model. Each of these SNMs would correspond to a state in the Kripke model.

The semantics of the privacy policy language $\mathcal{PPL}$ (included in $\mathcal{PPF}$) is given in terms of the satisfaction relation of $\mathcal{KBL}$, so $\mathcal{PPL}$ conformance is reduced to $\mathcal{KBL}$ satisfaction. Thanks to our results we may check conformance of $\mathcal{PPL}$ policies by using existing model checkers for epistemic logic.

IV.A Appendix

IV.A.1 Axiomatisation KD4

<u>Axioms</u>

(A1) All (instances of) first-order tautologies

(A2) $(K_i\varphi \wedge K_i(\varphi \implies \psi)) \implies K_i\psi$

(D) $\neg K_i\bot$

(A4) $K_i\varphi \implies K_iK_i\varphi$

<u>Derivation rules</u>

(Modus Ponens) $\dfrac{\varphi \qquad \varphi \implies \psi}{\psi}$

(Necessitation) $\dfrac{\varphi}{K_i\varphi}$ where φ must be provable from no assumptions.

IV.A.2 Soundness Proofs

Soundness of axiom D

Lemma 1. *For all $SN \in \mathcal{SN}$ and some agent i the following holds:*

$$SN \vDash \neg K_i\bot.$$

Proof. Assume by contradiction that $SN \vDash K_i\bot$. By $\vDash$ it follows $KB_i \vdash \bot$. In order to derive $\bot$, it must be possible to derive $KB_i \vdash \varphi$ and $KB_i \vdash \neg\varphi$. From Def. 6 it follows that if $KB_i \vdash \varphi$ then $KB_i \nvdash \neg\varphi$, thus deriving a contradiction. □

Soundness of K-axiomatisation

Lemma 2. *For all formulae φ and ψ in $\mathcal{F}_{\mathcal{KBL}}$, $SN \in \mathcal{SN}$ and some agent i the following holds:*

a) A1. All (instances of) first-order tautologies,

b) A2. $SN \vDash (K_i\varphi \wedge K_i(\varphi \implies \psi)) \implies K_i\psi$,

c) R1. From $SN \vDash \varphi$ and $SN \vDash \varphi \implies \psi$ infer $SN \vDash \psi$,

d) R2. From $\vDash \varphi$ infer $\vDash K_i\varphi$ where φ must be provable from no assumptions.

Proof.

a) It follows from the definition of the environment's knowledge base. KB_e is defined to include all truth, which includes all tautologies of FOEL.

b) It trivially follows since $\vdash$ includes the rule *modus ponens*.

c) It follows immediately from the fact that the interpretation of $\wedge$ and $\neg$ in the definition of $\models$ is the same to that of in First-Order Logic.

d) Let φ be a formula provable from no assumptions, i.e., $\models \varphi$. By axiom A1 it also holds that $KB_i \vdash \varphi$ for all KB_i. By $\models$ it follows $\models K_i\varphi$ for all $i \in Ag$ as required.

$\square$

Soundness of axioms A4 and A5

Lemma 3. *For all formula φ in $\mathcal{F}_{\mathcal{KBL}}$, $SN \in \mathcal{SN}$ and some agent i the following holds: A4. $SN \models K_i\varphi \implies K_iK_i\varphi$.*

Proof. It follows from Def. 7. Consider an agent i that knows a formula φ, i.e., $K_i\varphi$. Then $KB_i \vdash \varphi$. By SA (Def. 7) it follows $KB_i \vdash K_i\varphi$. Finally, by $\models$ it follows $SN \models K_iK_i\varphi$ as required. $\square$

Lemma 4. *For all formula φ in $\mathcal{F}_{\mathcal{KBL}}$, $SN \in \mathcal{SN}$ and some agent i the following holds: A5. $SN \not\models \neg K_i\varphi \implies K_i\neg K_i\varphi$.*

Proof. Here we provide a counter example for A5. Consider an empty knowledge base $KB_i = \emptyset$ for an agent $i \in Ag$. Let φ be a formula and SN the SNM consisting of the knowledge base of i. Since $SN \models \neg K_i\varphi$ holds iff $KB_i \not\vdash \varphi$, and $KB_i = \emptyset$, it follows (By Def. $\models$) that $SN \models \neg K_i\varphi$. However, $SN \models K_i\neg K_i\varphi$ holds iff $KB_i \vdash \neg K_i\varphi$. $\emptyset \not\vdash \neg K_i\varphi$ since $\neg K_i\varphi$ is not probable from no assumptions. Therefore, $SN \not\models K_i\neg K_i\varphi$ as required. $\square$

Soundness of KD45-axiomatisation

Theorem 2. *The **KD4** axiomatisation is sound with respect to SNMs.*

Proof. It follows from lemmas 1, 2 and 3. $\square$

Soundness of the axiomatisation of common knowledge

Lemma 5. *For all formula φ written in $\mathcal{F}_{\mathcal{KBL}}$, $SN \in \mathcal{SN}$ and some group of agents G the following holds:*

a) C1. $SN \models E_G\varphi \iff \bigwedge_{i\in G} K_i\varphi$,

b) C2. $SN \models C_G\varphi \iff E_G(\varphi \wedge C_G\varphi)$,

c) RC1. From $\models \varphi \implies E_G(\varphi \wedge \psi)$ infer $\models \varphi \implies C_G\psi$ where $\varphi \implies E_G(\psi \wedge \varphi)$ must be provable from no assumptions.

Proof.

a) It trivally follows from the definition of E_G.

b) It follows from the definition of common knowledge. First we prove the implication from right to left. Assume that $SN \models C_G\varphi$. It means that $KB_i \vdash E_G\varphi \wedge E_GE_G\varphi \wedge E_GE_GE_G\varphi \wedge \ldots$ for all $i \in G$. By distributivity of E_G we can derive that $KB_i \vdash E_G(\varphi \wedge E_GE_G\varphi \wedge E_GE_G\varphi \wedge \ldots)$. Hence $KB_i \vdash \varphi$ and $KB_i \vdash E_G\varphi \wedge E_GE_G\varphi \wedge \ldots$. Finally, by the definition of C_G, we conclude that $KB_i \vdash \varphi \wedge C_G\varphi$ for all $i \in G$, thus $SN \models E_G(\varphi \wedge C_G\varphi)$. The other direction of the implication follows directly from the definition of K_i and C_G. Assume $SN \models E_G(\varphi \wedge C_G\varphi)$. It means that for all $i \in G$, $KB_i \vdash C_G\varphi$, hence by the definition of C_G we know that $KB_i \vdash \varphi \wedge E_G\varphi \wedge E_GE_G\varphi \wedge \ldots$. By definition of $\models$ it follows $SN \models E_G(\varphi \wedge E_G\varphi \wedge E_GE_G\varphi \wedge \ldots)$. Finally, by distributivity of E_G and definition of C_G we conclude that $SN \models C_G\varphi$ as required.

c) Let $\varphi \implies E_G(\varphi \wedge \psi)$ be probable from no assumptions. **KD4** also includes RC1, therefore, using $\vdash$, from $\varphi \implies E_G(\varphi \wedge \psi)$ it can be derived that $\models \varphi \implies C_G\psi$ in all KB_i. Finally, since we made no assumption of any concrete SNM, we conclude that $\models \varphi \implies C_G\psi$.

□

Soundness of the axiomatisation of distributed knowledge

Lemma 6. *For all formula φ in $\mathcal{F}_{\mathcal{KBL}}$, $SN \in \mathcal{SN}$ and some group of agents G, the following holds:*

a) D1. $SN \models D_{\{i\}}\varphi \iff K_i\varphi,\ i = 1, \ldots, n$, *b) D2. $SN \vDash D_G\varphi \implies D_{G'}(\varphi)$ if $G \subseteq G'$,*

*c) DA2 and DA4. Axioms A2 and A4 of **KD4**, replacing the modality K_i with the modality D_G for each axiom.*

Proof.

a) It easily follows from the definition of distributed knowledge. $SN \vDash D_{\{i\}}\varphi$ iff $\bigcup_{j\in\{i\}} KB_j \vdash \varphi$. Also we have that $\bigcup_{j\in\{i\}} KB_j = KB_i$. Hence we can conclude that $D_{\{i\}}\varphi$ iff $KB_i \vdash \varphi$. By definition of $\vDash$, we know that $K_i\varphi$ iff $KB_i \vdash \varphi$. Therefore, we can conclude that $SN \vDash D_{\{i\}}\varphi \iff K_i\varphi$.

b) By the definition of distributed knowledge we know that $SN \vDash D_{G'}\varphi$ holds if and only if $\cup_{i\in G'} KB_i \vdash \varphi$, since $G \subseteq G'$, we can rewrite the previous expression as $(\cup_{i\in G} KB_i) \cup (\cup_{i\in G'\setminus G} KB_i) \vdash \varphi$. We know from the assumption that $D_G\varphi$ holds, hence $\cup_{i\in G} KB_i \vdash \varphi$. By Lemma 8, we can deduce that $(\cup_{i\in G} KB_i) \cup (\cup_{i\in G'\setminus G} KB_i) \vdash \varphi$ holds. Finally, by the definition of $\vDash$, we can conclude that $SN \vDash D_{G'}\varphi$.

c) Distributed knowledge is equivalent to an agent which contains all the knokwledge of the members of the group. Therefore, the proofs of all the axioms and derivation rules is analogous to those of Lemmas 2 and 3, just by replacing K_i with D_G.

□

Lemma 8. *Given the set formulae $\Phi \subseteq \mathcal{F}_{\mathcal{KBL}}$, two formulae $\varphi, \psi \in \mathcal{F}_{\mathcal{KBL}}$ the following holds:*

$$\textit{If } \Phi \vdash \varphi \textit{ then } \Phi \cup \{\psi\} \vdash \varphi.$$

Proof. Assume Φ or $\Phi \cup \{\psi\}$ are inconsistent, then anything can be derived from $\vdash$ (including φ), therefore if $\Phi \cup \{\psi\}$ is inconsistent, $\Phi \cup \{\psi\} \vdash \varphi$.

Assume now $\Phi \cup \{\psi\}$ is consistent. None of the axioms in $\vdash$ (i.e., **KD4**) remove any formula (see [29]). Therefore, adding more formula to the set of premises will strictly increase the number of formulae that can be derived. Thus, if $\Phi \vdash \varphi$ then we can conclude that $\Phi \cup \{\psi\} \vdash \varphi$. □

IV.A.3 Preservation of satisfiability of a canonical Kripke model constructed from an SNM

Theorem 3. *If a formula φ is satisfiable in an SNM SN then φ is satisfiable in the Kripke model $\mathcal{KT}(SN)$.*

Proof. Fagin *et al.* have shown that a canonical Kripke model which satisfies a **KD4**-consistent formula (and everything that can be derived from it) can be constructed [29]. Their proof can be adapted to our case. In particular, we show that if a formula φ is satisfiable in *SN*, then it is included in a maximal **KD4**-consistent set. Let $\mathit{Con}(\Phi_{SN})$ be a set containing all **KD4** consistent sets generated from all subformulae of Φ_{SN} and their negations (see [29, Theorem 3.2.4] for the formal definition of how to construct $\mathit{Con}(\Phi_{SN})$). We will prove that for any formula φ, if $SN \models \varphi$ then $\varphi \in V$ where $V \in \mathit{Con}(\Phi_{SN})$ and by [29, Theorem 3.2.4] it holds that φ is satisfiable in the corresponding canonical Kripke model defined in Def. 14. The proof is carried out by induction on the structure of the formula.

Case $\varphi = p(\overrightarrow{t})$. Assume that $SN \models p(\overrightarrow{t})$ holds, then $p(\overrightarrow{t}) \in KB_e$ and by Def. 14 $p(\overrightarrow{t}) \in \Phi_{SN}$. Since Φ_{SN} is a **KD4** consistent set, there exists a $V \in \mathit{Con}(\Phi_{SN})$ such that $V = \Phi_{SN}$.

Case $\varphi = a(i,j)$ or $\varphi = c(i,j)$. Assume that $SN \models a(i,j)$ holds, then $a(i,j) \in A_a$ and by Def. 14 $a(\overrightarrow{t}) \in \Phi_{SN}$. For the same reason as before, there exists a $V \in \mathit{Con}(\Phi_{SN})$ such that $V = \Phi_{SN}$. The exact same proof holds for $c(i,j)$.

Case $\varphi = \neg\psi$. By induction hypothesis assume that $\psi \in V$ for some $V \in \mathit{Con}(\Phi_{SN})$. Since ψ is a subformula of φ, from the construction of $\mathit{Con}(\Phi_{SN})$ there exists a set V such that $\neg\psi \in V$.

Case $\varphi = \psi_1 \wedge \psi_2$. Since φ is **KD4** consistent there exists a set $V \in \mathit{Con}(\Phi_{SN})$ such that $\varphi \in V$.

Case $\varphi = \forall x.\psi$. By induction hypothesis assume that if $SN \models \psi[v/x]$ then $\psi[v/x] \in V$ for some $V \in \mathit{Con}(\Phi_{SN})$. By definition of $\models$ it follows that $SN \models \forall x.\psi$ iff $SN \models \psi[v/x]$ for all $v \in D_o$, hence concluding that $\psi[v/x] \in V$.

Case $\varphi = K_i\psi$. Assume that $SN \models K_i\psi$. By definition of $\models$ it holds that $KB_i \vdash \psi$. It leads to two possible cases: 1) If $K_i\psi \in \Phi_{SN}$ then $\psi \in V$ for $V \in \mathit{Con}(\Phi_{SN})$ such that $V = \Phi_{SN}$. 2) Otherwise, ψ has been derived, i.e., $KB_i \vdash \psi$, since $\vdash$ and **KD4** include the same axioms and derivation rules, there exists a state $V \in \mathit{Con}(\Phi_{SN})$ which includes $\varphi_{SN} \in V$ and also $K_i\psi \in V$ is derived.

Case $\varphi = S_G\psi$ or $\varphi = E_G\psi$. Both are derived operators from K_i. The proof easily follows by their definition and the proof provided for the case $\varphi = K_i\psi$.

Case $\varphi = C_G\psi$. Assume that $SN \models C_G\psi$ holds. By axiom C2 and the definition $\models$ we can derive that $\forall j \in G$ $KB_j \vdash (\psi \wedge C_G\psi)$. Thus, there exists $V \in \mathit{Con}(\Phi_{SN})$ such that $E_G(\psi \wedge C_G\psi) \in V$. Applying again C2 in V we can derive that $C_G\psi \in V$ as required.

Case $\varphi = D_G\psi$. Let $DKB_G \triangleq \cup_{i \in G} KB_i$. Assume that $SN \vDash D_G\psi$. By definition of $\vDash$ it holds that $DKB_G \vdash \psi$. It leads to two possible cases: 1) If $\psi \in DKB_G$ then there exist an agent $i \in G$ such that $\psi \in KB_i$ therefore $K_i\psi \in \Phi_{SN}$ by Def. 14. Consequently, $K_i\psi \in V$ for $V \in Con(\Phi_{SN})$ such that $V = \Phi_{SN}$. Applying axioms D1 and D2 in V, we conclude that $D_G\psi \in V$. 2) Otherwise, ψ is derived from the explicit knowledge of the agents using any of the axioms in **KD4** (including common knowledge and distributed knowledge). Therefore we can conclude that $\psi \in V$ for $V \in Con(\Phi_{SN})$ such that V includes the conjunction of the knowledge of all agents in G.

□

Consistency of the characteristic formula φ_{SN}

Lemma 6. *For all $SN \in \mathcal{SN}$, φ_{SN} is **KD45**-consistent.*

Proof. No negated connection or action predicates are added to Φ_{SN}. Therefore, no inconsistency can be derived from $\{c(i,j) \mid (i,j) \in C_c, c \in \mathcal{C}\} \cup \{a(i,j) \mid (i,j) \in A_a, a \in \Sigma\} \cup \{p(\overrightarrow{t}) \mid p(\overrightarrow{t}) \in KB_e\}$. Since **KD4** does not include the axiom $K_i\varphi \implies \varphi$ the knowledge of the agents will not make Φ inconsistent. Moreover, the knowledge bases of the agents are assumed to be consistent Def. 6, in particular we assume that agents can derive new knowledge according the axiomatisation **KD4**. Only what the agents know is included in Φ_{SN}, i.e., Φ_{SN} does not contain any formula of the form $\neg K_i\psi$, therefore no contradiction can be derived. Given the above it follows that φ is consistent. □

IV.A.4 Preservation of satisfiability of an SNM constructed from a canonical Kripke model

We prove that all formulae that hold in the state of the canonical Kripke model corresponding to the characteristic formula are satisfiable in the original SNM. Here, we also provide all the formal definitions required for the proof.

Definition 15 (Marked characteristic set)**.** *The* marked characteristic set *of an SNM SN, denoted as Φ^m_{SN}, is constructed as follows:*

$$\begin{aligned} \Phi^m_{SN} = \{pr\_p(\overrightarrow{t}) \mid p(\overrightarrow{t}) \in KB_e\} \cup \\ \{co\_c(i,j) \mid (i,j) \in C_c, c \in \mathcal{C}\} \cup \\ \{ac\_a(i,j) \mid (i,j) \in A_a, a \in \Sigma\} \cup \\ \{K_i\varphi \mid \varphi \in KB_i\} \end{aligned}$$

Definition 16 (Marked Kripke transformation function)**.** *We define the* marked Kripke transformation function*, denoted $\mathcal{KT}^m$, as $\mathcal{KT}$ in Def. 14 except for step 1), where $\mathcal{KT}^m$ constructs a marked characteristic set (as defined in Def. 15).*

Theorem 4. *If a formula φ is satisfiable in the state $s_{\Phi^m_{SN}}$ of a Kripke model $\mathcal{KT}^m(SN)$ then φ is satisfiable in the SNM SN.*

Proof. As we mentioned in Theorem 3, Fagin *et al.* show that all formula φ is satisfiable in a state s_Φ iff $\varphi \in V$ where V is a maximal **KD4** consistent set. We refer the reader to [29, Theorem 3.2.4] for the formal details of how to extend a **KD4** consistent set to a maximal **KD4** consistent set. Let V be a maximal **KD4** consistent set from Φ^m_{SN}. We can reduce the proof of this theorem to showing that if $\varphi \in V$ then $SN \vDash \varphi$. The proof is split in two cases: 1) $\varphi \in \Phi^m_{SN}$ and 2) $\varphi \in V \setminus \Phi^m_{SN}$.

Case $\varphi \in \Phi^m_{SN}$. We split the proof in the three possible formulas that can be included in Φ^m_{SN} by the definition of 16:

- Case $\varphi = pr\_p(\vec{t})$. Let $pr\_p(\vec{t}) \in \Phi^m_{SN}$. By Def. 16 we derive that $p(\vec{t}) \in KB_e$ and by $\vDash$ we conclude that $SN \vDash p(\vec{t})$ holds.
- Case $\varphi = ac\_a(i,j)$ or $\varphi = a(i,j)$. Let $ac\_a(i,j) \in \Phi^m_{SN}$. By Def. 16 we derive that $p(\vec{t}) \in A_a$ and by $\vDash$ we conclude that $SN \vDash a(i,j)$ holds. The same reasoning holds for $co\_c(i,j)$.
- Case $\varphi = K_i\psi$. Let $K_i\psi \in \Phi^m_{SN}$. By Def. 16 we derive that $\psi \in KB_i$ and by $\vDash$ we conclude that $SN \vDash K_i\psi$ holds.

Case $\varphi \in V \setminus \Phi^m_{SN}$. If a formula φ is in $\varphi \in V \setminus \Phi^m_{SN}$ it means that it is has been derived from the explicit knowledge and predicates present in Φ^m_{SN}. As we have shown in Section IV.4, the axiomatisation **KD4** is sound. Therefore the same derivations can be performed in *SN*.

□

In general, all elements in the a marked characteristic set can be uniquely identified in an SNM, thus it is possible transform a marked characteristic set to the corresponding SNM.

Definition 17 (SNM construction)**.** *Let $\mathcal{UMS}$ be the the universe of all possible marked characteristic sets. We define the* SNM construction *function, $\mathcal{SC} : \mathcal{UMS} \to \mathcal{SN}$ as follows: The resulting SNM for Φ^m_{SN}:*

$$SN = \langle Ag, \mathcal{A}, KB, \pi \rangle$$

where

- $Ag = \{i \mid K_i\varphi \in Sub(\varphi^m_{SN})\}$
- $\mathcal{A}$ *contains all function symbols, constant symbols and relation symbols (without the marking) present in the formulae of* Φ^m_{SN}*.*
  - $KB_e = \{p(\overrightarrow{t}) \mid pr\_p(\overrightarrow{t}) \in \Phi^m_{SN}\}$
  - *for all* $c \in \mathcal{C}$*,* $C_c = \{(i,j) \mid co\_c(i,j) \in \Phi^m_{SN}\}$
  - *for all* $a \in \Sigma$*,* $A_a = \{(i,j) \mid ac\_a(i,j) \in \Phi^m_{SN}\}$
- $KB = \{KB_i = \{\varphi \mid K_i\varphi \in \Phi^m_{SN}\}\}_{i \in Ag}$
- $\pi = \emptyset$[4]

IV.A.5 Comparison of the complexity of the satisfiability problem in SNMs and Kripke models

Lemma 7. *Given* $SN \in \mathcal{SN}$ *and a formula* φ *the following holds*

$$O(\sum_{K_i\varphi_i \in outerK(\varphi)} (2^{|\varphi_{KB_i}|} \times |\varphi_i|)) < O(2^{|\varphi_{SN}|} \times |\varphi^K|).$$

Proof. The formula φ^K is a conjunction of formulae starting with a K_i modality. More specifically, φ^K has the following shape

$$\varphi^K = K_1\varphi^1_1 \wedge K_i\varphi^1_2 \wedge \ldots K_2\varphi^2_1 \wedge K_2\varphi^2_2 \wedge \ldots \wedge K_m\varphi^m_n$$

where $m \in Ag$ and $n \in \mathbb{N}$. Since $K_i\varphi \wedge K_i\psi \implies K_i\varphi \wedge \psi$ holds for any φ and ψ, φ^K can always be rewritten as

$$\varphi^K = K_1\varphi_1 \wedge \ldots \wedge K_m\varphi_m$$

where $\varphi_1 = \varphi^1_1 \wedge \varphi^1_2 \wedge \ldots$ and $\varphi_m = \varphi^m_1 \wedge \varphi^m_2 \wedge \ldots$. Since all φ_m are subformulae of φ^K (without their corresponding K_i modality) the sum of their lenghts will always be strictly smaller, i.e.,

$$|\varphi_1| + \ldots + |\varphi_i| < |\varphi^K|$$

Given the above, the following trivially holds

[4] π is empty since Kripke models do not contain information about the privacy policies of the agents.

$$O(|\varphi_1| + \ldots + |\varphi_i|) < O(|\varphi^K|)$$

Multiplying by a constant c in both sides of the inequality does not affect the result

$$O(c \times (|\varphi_1| + \ldots + |\varphi_i|)) < O(c \times |\varphi^K|)$$

By replacing c with $2^{|\varphi_{SN}|}$ in the previous statement we obtain the following

$$O(2^{|\varphi_{SN}|} \times (|\varphi_1| + \ldots + |\varphi_i|)) < O(2^{|\varphi_{SN}|} \times |\varphi^K|)$$

Since multiplication distributes over addition the following holds

$$O(2^{|\varphi_{SN}|} \times |\varphi_1| + \ldots + 2^{|\varphi_{SN}|} \times |\varphi_i|) < O(2^{|\varphi_{SN}|} \times |\varphi^K|) \qquad \text{(IV.1)}$$

As we describe in Section IV.6 φ_{KB_i} represents the conjunction of all formulae in the knowledge base of agent i. Besides, the characteristic formula φ_{SN} is a conjunction of the formulae of the knowledge bases of all agents plus the normal predicates. Therefore for any $i \in Ag$ it holds that $2^{|\varphi_{KB_i}|} < 2^{|\varphi_{SN}|}$. From this fact and (IV.1) we derive the following

$$O(2^{|\varphi_{KB_1}|} \times |\varphi_1| + \ldots + 2^{|\varphi_{KB_i}|} \times |\varphi_i|) < O(2^{|\varphi_{SN}|} \times |\varphi_1| + \ldots + 2^{|\varphi_{SN}|} \times |\varphi_i|) \quad \text{(IV.2)}$$

Finally, by transitivity of $<$ in (IV.1) and (IV.2) we conclude that

$$O(2^{|\varphi_{KB_1}|} \times |\varphi_1| + \ldots + 2^{|\varphi_{KB_i}|} \times |\varphi_i|) < O(2^{|\varphi_{SN}|} \times |\varphi^K|)$$

as required. $\square$

Chapter V

An Automata-based Approach to Evolving Privacy Policies for Social Networks

Raúl Pardo, Christian Colombo, Gordon J. Pace and Gerardo Schneider

Abstract. *Online Social Networks* (OSNs) are ubiquitous, with more than 70% of Internet users being active users of such networking services. This widespread use of OSNs brings with it big threats and challenges, privacy being one of them. Most OSNs today offer a limited set of (static) privacy settings and do not allow for the definition, even less enforcement, of more dynamic privacy policies. In this paper we are concerned with the specification and enforcement of *dynamic* (and *recurrent*) privacy policies that are activated or deactivated by context (events). In particular, we present a novel formalism of *policy automata*, transition systems where privacy policies may be defined per state. We further propose an approach based on runtime verification techniques to define and enforce such policies. We provide a proof-of-concept implementation for the distributed social network Diaspora, using the runtime verification tool Larva to synthesise enforcement monitors.

V.1 Introduction

As stated in [96] by Weitzner et al, "[p]rotecting privacy is more challenging than ever due to the proliferation of personal information on the Web and the increasing analytical power available to large institutions (and to everyone else) through Web search engines and other facilities". The problem being not only to determine *who* might be able to access *what* information and *when* but also *how* the information is going to be used (for which *purpose*). Addressing all these privacy-related questions is complex, and as today there is no ultimate solution.

The above is particularly true for *Online Social Networks* (OSNs) (also known as *Social Networking Sites* or *Social Networking Services* —SNSs), due to their explosion in popularity in the last years. Sites like Facebook, Twitter and LinkedIn are in the top 20 most visited Web sites in the world [2]. Nearly 70% of the Internet users are active on OSNs as shown in a recent survey [45], and this number is increasing. A number of studies show that the number of privacy breaches is keeping pace with this growth [56, 41, 51, 57]. The reasons for this increase on privacy breaches are manifold; just to mention a few: i) Many users are not aware of the implications of content sharing on OSNs, and do not foresee the consequences until it is too late; ii) Most users do not take the time to check/change the default privacy settings, which are usually quite permissive; iii) The privacy settings offered by existing OSNs are limited and are not fine-grained enough to capture desirable privacy policies; iv) Side knowledge and indirect disclosure, e.g. through aggregation of information from different sources, it is difficult to foresee and detect; v) There currently are no good warning mechanisms informing users of the potential breach of privacy, before a given action is taken; vi) Privacy settings are static (they are not time- nor context-dependent), thus not being able to capture the possibility of defining repetitive or recurrent privacy policies.

Recently, the following privacy flaw was pointed out in the Facebook messenger app [9]. It was shown that it is possible to track users based on their previous conversations. It was enough to chat several times per day with users to accurately track their locations and even infer their daily routines. It was possible since the app adds by default the location of the sender to all the messages. This problem arises because of some of the reasons in the previous list such as i), ii) and v). Facebook solution was to disable location sharing by default, which might be seen as a too radical solution. However, it is the best Facebook developers can do given the current state of privacy protection mechanisms. We believe that there is room for better solutions that offer protection to users while not restricting the sharing functionalities of the OSN. For instance, this privacy flaw could have been solved with a privacy policy that says *"My location can only be disclosed 3 times per day"*. This policy prevents tracking users while still allowing users to share their location in a controlled manner. We called this type of privacy policies *evolving* polices and they are the focus of this paper. Other examples of evolving policies are *"Co-workers cannot see my posts while I am not at work, and only family can see my location while I am at home"* or *"My supervisor cannot see my pictures during the weekend"*.

In this paper we address the above problem, through the following contributions: i) The definition of *policy automata* (finite state automata enriched with privacy policies in their states), the definition of a subsumption and a conflict relation between policy automata, and the proofs of some properties about these relations (Section V.2); ii) A translation from policy automata into DATEs [17], the underlying data structure of the runtime verification tool LARVA [18] (Section V.3); iii) A proof-of-concept implementation of dynamic/recurrent privacy policies for the open source distributed OSN Diaspora* [22] using LARVA (Sections V.4,V.5).

V.2 Policy automata

In order to describe evolving policies, we adopt the approach of taking a static policy language and use it to describe temporal snapshots of the policies in force. We then use a graph structure to describe how a policy is discarded and another enforced, depending on the events taking place e.g. user actions or system events.

V.2.1 Semantics of Policy Automata

Policy automata are defined as structures such that progressing through structure represents evolving policies, parametrised by a static policy language SPL. This approach allows us to define a whole family of evolving policy languages, depending on the underlying static language used.

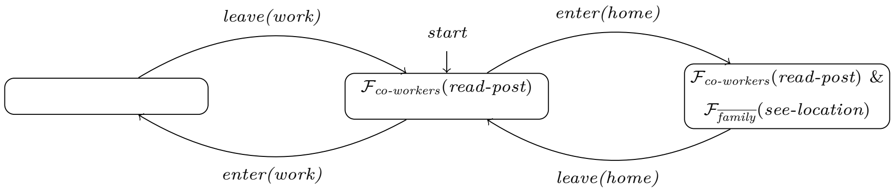
Assumption 1. *We assume that SPL has the notion of conjunction of policies such that, for any two policies*[1] $p_1, p_2 \in SPL$, $p_1 \,\&\, p_2 \in SPL$.

Definition 1. *A* policy automaton *over a static privacy policy language SPL is a 4-tuple* $\langle \Sigma, Q, q_0, \rightarrow, \pi \rangle$ *where:* Σ *is the alphabet — effectively the set of observable actions of the underlying system;* Q *is the set of states in the automaton;* $q_0 \in Q$ *is the initial state of the automaton;* $\rightarrow \subseteq Q \times \Sigma \times Q$ *is the transition relation; and* $\pi \in Q \rightarrow SPL$ *is a function which maps each state to a privacy policy in SPL.*

[1]In the rest of the paper we take SPL to be the set of well-formed policy formulae of the static policy language.

We will write $q \xrightarrow{a} q'$ to indicate that there is a transition from state q to state q', labelled by a: $q \xrightarrow{a} q' \overset{df}{=} (q, a, q') \in \rightarrow$. We will take the transitive closure of the transition relation to enable us to write $q \overset{es}{\Longrightarrow} q'$ to denote that the sequence of events es takes the automaton from state q to state q'.

Example 1. *To illustrate policy automata let us consider the policy* 'Co-workers cannot see my posts while I am not at work, and only family can see my location while I am at home' *(P1). If we use the static policy operator $\mathcal{F}_g(x)$ to denote that anyone in group g is forbidden from performing action x (where x can refer to posting, viewing a post, liking a post, etc.), we can express the first part of P1 to be $\mathcal{F}_{\textit{co-workers}}(\textit{read-post})$, and the second part to be $\mathcal{F}_{\overline{\textit{family}}}(\textit{see-location})$ (we use $\bar{g}$ to denote the complement of a group of users g). By synchronising with the actions of our social network application through events marking the arrival at and departure from a location (enter(l) and leave(l) respectively), we can express the evolving policy in the following manner[2]:*


leave(work)
start
enter(home)
$\mathcal{F}_{\textit{co-workers}}(\textit{read-post})$
$\mathcal{F}_{\textit{co-workers}}(\textit{read-post})$ &
$\mathcal{F}_{\overline{\textit{family}}}(\textit{see-location})$
enter(work)
leave(home)

Non-deterministic and non-total transition relations in a policy automaton can lead to policy behaviour which is typically not required in real-life policy analysis. For instance, we do not want to consider automata that under the execution of an event, randomly choose between the activation of two different static policies. For this reason, we define the subset of sane policy automata which behave deterministically and never deadlock.

Definition 2. *We say that a policy automaton $\mathcal{P} = \langle \Sigma, Q, q_0, \rightarrow, \pi \rangle$ is* sane *if its transition relation is total and deterministic (functional). With sane policies, we write $q \xrightarrow{e}$ and $q \overset{es}{\Longrightarrow}$ (with $e \in \Sigma$ and $es \in \Sigma^*$) to denote the unique state reachable from state q, following action e and sequence es respectively. Finally, we will write $policy_{\mathcal{P}}(es)$ to denote the policy in force after following event sequence es from the initial state: $policy_{\mathcal{P}}(es) \overset{df}{=} \pi(q_0 \overset{es}{\Longrightarrow})$.*

In order to give a semantics to policy automata, we require the semantics of the underlying static policy language. Let $\sigma \in SN$ be the state of the social network where

[2]When we draw a policy automaton, transitions for events that are not explicitly drawn are assumed to be reflexive.

SN is the universe of all possible social network states. Given a static policy language SPL, we write $\sigma,\ e \vdash_{SPL} p$ to denote that in the social network state σ an event e respects privacy policy p. We assume that the social network (but not the policy) may evolve over time through events via the relation $\rightarrow_{SN} \subseteq SN \times \Sigma \times SN$ which is assumed to be a total function on the two first parameters.

Based on the semantics of the static policy language, we can now define the semantics of policy automata:

Definition 3. *The* configuration *of a policy automaton consists of the state of the automaton*[3]. *The initial configuration is taken to be* q_0. *Whether an* event respects a policy automaton in a particular configuration C *is defined as follows:*

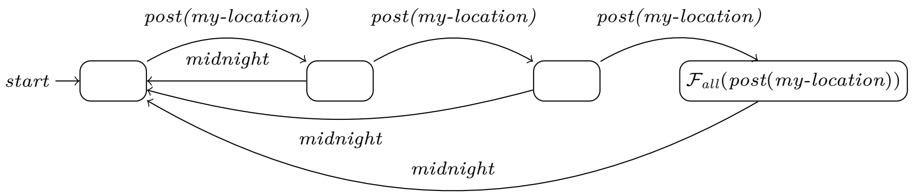
$$\frac{\sigma,\ e \vdash_{SPL} \pi(C)}{\sigma,\ e \vdash_{PA} C}\ \text{SPL}$$

This is extended over traces in the following manner:

$$\frac{}{\sigma,\ \varepsilon \vdash_{PA} C}\ \textsc{BaseTrace}$$

$$\frac{\sigma,\ e \vdash_{PA} C \qquad \sigma \xrightarrow{e}_{SN} \sigma' \qquad C \xrightarrow{e} C' \qquad \sigma',\ es \vdash_{PA} C'}{\sigma,\ e : es \vdash_{PA} C}\ \textsc{IndTrace}$$

Example 2. *Consider the policy* 'Only up to 3 posts disclosing my location are allowed per day in my timeline' *(P2), which can be encoded as the following automaton (we will assume that from left to right, the states are named* q_0, q_1, q_2 *and* q_3*):*



Since we expect that posting the location when a policy prohibiting it is in force is a violation, we would expect the static policy language semantics to show that for any social network state σ*:* $\sigma,\ post(my\text{-}location) \not\vdash_{SPL} \mathcal{F}_{all}(post(my\text{-}location))$.

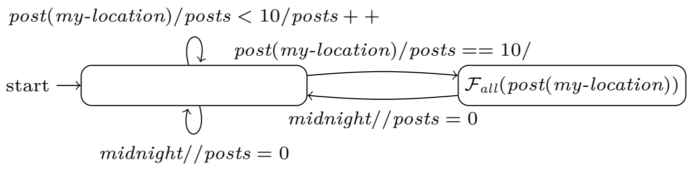
[3]We present these semantics in terms of general configurations, rather than the automata states, since we envisage the extension of the automata to handle local symbolic state, requiring a richer configuration but still in line with the definitions given in this paper.

From this, and given that $\pi(q_3) = \mathcal{F}_{all}(post(my\text{-}location))$ we can deduce that in state q_3, the policy clause is likewise violated whenever a post disclosing my-location is performed, no matter the state of the social network: σ, $post(my\text{-}location) \not\vdash_{PA} q_3$.

Using the rule IndTrace*, provided there is σ' such that $\sigma \xRightarrow{post(my\text{-}location)^3} \sigma'$, we have:*[4] σ, $post(my\text{-}location)^4 \not\vdash_{PA} q_0$.

Note that here we write $post(my\text{-}location)^4$ because we want to check that after disclosing 3 times the user's location, the forth one would be a violation of $\pi(q_3)$.

If the maximum number of posts were to be increased, the number of states in the automaton would grow quickly. For the sake of presentation, in the rest of the paper, we will also be enriching our notation in the examples to transition systems which have an implicit symbolic state. Transitions are labelled by a triple: *event*/*condition*/*state-update* — triggering when the specified event happens and the condition holds, performing the state update before proceeding. The property allowing for 10 location posts can be expressed in this notation in the following manner:


$post(my\text{-}location)/posts < 10/posts++$
$post(my\text{-}location)/posts == 10/$
start
$\mathcal{F}_{all}(post(my\text{-}location))$
$midnight//posts = 0$
$midnight//posts = 0$

Such a symbolic automaton can be unfolded into a policy automaton possibly with an infinite number of states. For instance, in the above example, the set of states would be $\{(q, n) \mid q \in \{q_0, q_1\},\ n \in \mathbb{N}\}$ where q holds the value of the (explicit) state, and n the value of *posts*. Since in this paper we are concerned with runtime verification — enforcing a dynamic policy along a single trace, the infinite number of states poses no challenge to the decidability question.

States in policy automata do not contain all the privacy policies which are being enforced in the OSN. Internally the OSN could be enforcing other static policies that have been manually activated by the users. Policy automata are a separate layer to control some static policies. When a policy automaton moves to a state, the static policies in the new state are activated in the OSN. Similarly, when the automaton leaves a state, the static polices are deactivated. Transitions to and from an empty state just mean that there is no update of static policies.

One advantage of using policy automata is that one can combine them synchronously to get the equivalent of conjunction over evolving policies. In order to do so, we require the underlying SPL to have a notion of conjunction (cf. Assumption 1).

[4]The supra-index over events represent the number of occurrences of the event, so *my-location*^{3} represent the sequence of events *my-location*; *my-location*; *my-location*.

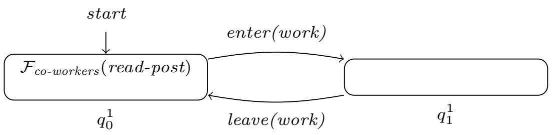
Policy automata can now be combined using standard synchronous composition over a particular alphabet:

Definition 4. *Given two policy automata $\mathcal{P}_1$ and $\mathcal{P}_2$ (such that $\mathcal{P}_i = \langle \Sigma_i, Q_i, q_{0i}, \rightarrow_i, \pi_i \rangle$), the synchronous composition of the automata synchronising over actions G, is defined to be the policy automaton $\mathcal{P}_1 \|_G \mathcal{P}_2 = \langle \Sigma_1 \cup \Sigma_2,\ Q_1 \times Q_2,\ (q_{01}, q_{02}),\ \rightarrow, \pi \rangle$ where $\pi(q_1, q_2) \overset{df}{=} \pi_1(q_1) \,\&\, \pi_2(q_2)$ and the transition relation is defined as follows:*

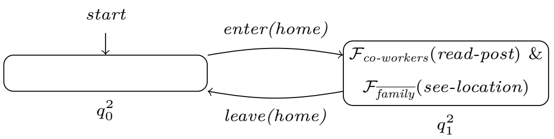
$$\frac{q_1 \xrightarrow{a}_1 q_1' \qquad q_2 \xrightarrow{a}_2 q_2'}{(q_1, q_2) \xrightarrow{a} (q_1', q_2')}\ a \in G$$

$$\frac{q_1 \xrightarrow{a}_1 q_1'}{(q_1, q_2) \xrightarrow{a} (q_1', q_2)}\ a \notin G \qquad\qquad \frac{q_2 \xrightarrow{a}_2 q_2'}{(q_1, q_2) \xrightarrow{a} (q_1, q_2')}\ a \notin G$$

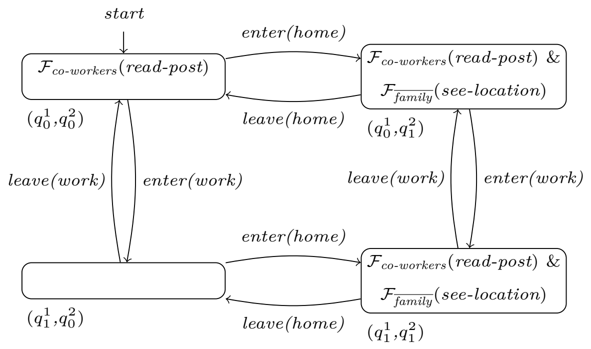
Example 3. *The policy automaton of Example 1 effectively is a composition of two individual evolving policies. First* "Colleagues cannot see my posts when I am not at work", *which can be represented in the following automaton*



and secondly, "Only my family can see my location while I am at home"*:*


start
enter(home)
$\mathcal{F}_{co\text{-}workers}(read\text{-}post)$ &
$\mathcal{F}_{\overline{family}}(see\text{-}location)$
leave(home)
q_0^2
q_1^2

Let $\mathcal{P}_1$ and $\mathcal{P}_2$ denote the previous two automata, respectively. The following diagram shows $\mathcal{P}_{12}$, the parallel composition of the previous automata $\mathcal{P}_1 \|_\emptyset \mathcal{P}_2$ (the synchronisation set is empty because $\mathcal{P}_1$ and $\mathcal{P}_2$ do not communicate over any event):



start
$\mathcal{F}_{\textit{co-workers}}(\textit{read-post})$
(q_0^1,q_0^2)
enter(home)
leave(home)
$\mathcal{F}_{\textit{co-workers}}(\textit{read-post})$ &
$\mathcal{F}_{\overline{\textit{family}}}(\textit{see-location})$
(q_0^1,q_1^2)
leave(work)
enter(work)
leave(work)
enter(work)
enter(home)
leave(home)
(q_1^1,q_0^2)
$\mathcal{F}_{\textit{co-workers}}(\textit{read-post})$ &
$\mathcal{F}_{\overline{\textit{family}}}(\textit{see-location})$
(q_1^1,q_1^2)

Note that this automaton is not equivalent to that of Example 1. In some transitions that Example 1's automaton do not update the static privacy policies (i.e., the automaton remains in the same state) this synchronous composition updates the policies accordingly. Imagine, for instance, that a user goes from work to home without leaving work (it is a possible scenario if the user lives at her workplace). After receiving enter(work), enter(home), the automaton resulting from the synchronous composition would active the policy $\mathcal{F}_{\textit{co-workers}}(\textit{read-post})$ *&* $\mathcal{F}_{\overline{\textit{family}}}(\textit{see-location})$ *whereas Example 1's automaton would activate no policies. Formally, the state* (q_0^1,q_1^2) *should contain the static policy* $\mathcal{F}_{\textit{co-workers}}(\textit{read-post})$ *&* $\mathcal{F}_{\textit{co-workers}}(\textit{read-post})$ *&* $\mathcal{F}_{\overline{\textit{family}}}(\textit{see-location})$*. However, we require the & operator of the static policy language to be idempotent (cf. Assumption 2, see below), thus being able to reduce the policy to* $\mathcal{F}_{\textit{co-workers}}(\textit{read-post})$ *&* $\mathcal{F}_{\overline{\textit{family}}}(\textit{see-location})$*.*

Though formally the evolving policies can thus be combined into a single one, in practice one can keep them separate and enforce them independently, e.g. possibly on separate machines, thus avoiding information leaks (if all the policies) have to be communicated to a central server for enforcement. For instance, one can see a user's set of policies being combined together over his or her local alphabet, and then synchronising globally at a global level across users:

$$(p_{1,1} \|_{U_1} \cdots \|_{U_1} p_{1,n}) \; \|_{\text{Global}} \; (p_{m,1} \|_{U_m} \cdots \|_{U_m} \; p_{m,n'})$$

V.2.2 Subsumption of dynamic privacy policies

Many notions can be carried over from the underlying static policy language to dynamic policies expressed using policy automata. Provided that the static policy language has a notion of semantic equivalence (which encompasses the usual properties of idempotency, commutativity and associativity of conjunction), we can derive equivalence and strictness ordering over policy automata.

Assumption 2. *We assume that the static policy language SPL has the notion of semantic equivalence $=_{SPL}$ which is assumed to be an equivalence relation.*

Furthermore, conjunction is assumed to be commutative, associative and idempotent under this equivalence: (i) $p_1\&p_2 =_{SPL} p_2\&p_1$; (ii) $p_1\&(p_2\&p_3) =_{SPL} (p_1\&p_2)\&p_3$; and (iii) $p\&p =_{SPL} p$.

Based on this equivalence, we can extend this to policy automata equivalence by quantifying over traces:

Definition 5. *Two policy automata $\mathcal{P}_1$ and $\mathcal{P}_2$ (with $\mathcal{P}_i = \langle \Sigma_i, Q_i, q_{0i}, \rightarrow_i, \pi_i \rangle$) with a common alphabet Σ (which requires $\Sigma_1 = \Sigma_2$) are equivalent if after following any trace, they both end up in a state in which the policies are equivalent:*

$$\mathcal{P}_1 =_{PA} \mathcal{P}_2 \overset{df}{=} \forall es : \Sigma^* \cdot policy_{\mathcal{P}_1}(es) =_{SPL} policy_{\mathcal{P}_2}(es)$$

Using standard approach, we can now define policy strictness ordering — a policy is considered stricter than another if all behaviour allowed by the former is also allowed by the latter.

Definition 6. *Given policy automata $\mathcal{P}_1$ and $\mathcal{P}_2$ over alphabet Σ, we say that $\mathcal{P}_1$* is stricter than *$\mathcal{P}_2$, written $\mathcal{P}_1 \sqsubseteq_{PA} \mathcal{P}_2$ as follows:*

$$\mathcal{P}_1 \sqsubseteq_{PA} \mathcal{P}_2 \overset{df}{=} \mathcal{P}_1 \|_{\Sigma} \mathcal{P}_2 =_{PA} \mathcal{P}_1$$

The strictness relation can be shown to obey certain properties.

Lemma 1. *The relation $\sqsubseteq_{PA}$ is transitive, antisymmetric and reflexive.*

Example 4. *Consider the policy automaton in Example 1 ($\mathcal{P}_1$) and the synchronous composition of the two automata in Example 3 ($\mathcal{P}_{12}$).*

As we remarked in Example 3, the two policy automata are clearly not equivalent. However, we would expect $\mathcal{P}_{12}$ to be a stricter version of $\mathcal{P}_1$. To show this, we note that the synchronous composition of $\mathcal{P}_1$ and $\mathcal{P}_{12}$, $\mathcal{P}_1 \|_{\Sigma} \mathcal{P}_{12}$ (where Σ is the whole alphabet, including $\{leave(home),\ leave(work),\ enter(home),\ enter(work)\}$), and $\mathcal{P}_{12}$ result in identical policies after following any trace. Formally, for all traces $es \in \Sigma^* \cdot policy_{\mathcal{P}_{12}\|\mathcal{P}_1}(es) =_{SPL} policy_{\mathcal{P}_{12}}(es)$, and thus we can conclude that $\mathcal{P}_{12}$ is stricter than $\mathcal{P}_1$: $\mathcal{P}_{12} \sqsubseteq_{PA} \mathcal{P}_1$.

V.2.3 Conflicting policy automata

In a similar manner as policy equivalence can be lifted from the static policy language to evolving policies, we can extend the notion of conflicting policies. Two static policies conflict when both cannot be satisfied or enforced at the same time. For example, imagine that Alice sets the policy *"Everyone can see the posts on my timeline"* and Bob activates a policy saying *"Only my friends can see my posts"*. If Bob posts in Alice's timeline which policy would apply? If the audience of the post is only Bob's friends Alice's policy would be violated. Similarly, if the audience of the posts is everyone, Bob's policy would not be satisfied. In order to define conflicting policy automata, we require the static policy language to include the notion of conflict between policies.

Assumption 3. *The static policy language SPL must be equipped with the notion of conflicting polices $\maltese_{SPL}$, which is assumed to be (i) symmetric; and (ii) closed under conjunction: if $p_1 \maltese_{SPL} p_2$ then for any p_1', it also holds that $(p_1 \,\&\, p_1') \maltese_{SPL} p_2$.*

We can lift the static policy conflict relation to one on evolving policies:

Definition 7. *Given any static policy language SPL and policy automata $\mathcal{P}_1$ and $\mathcal{P}_2$ with alphabet Σ:*

$$\mathcal{P}_1 \maltese_{PA} \mathcal{P}_2 \overset{df}{=} \exists es \in \Sigma^* \cdot policy_{\mathcal{P}_1}(es) \maltese_{SPL} policy_{\mathcal{P}_2}(es).$$

The intuition behind the previous definition is simple. Any two automata are in conflict if after the execution of a sequence of events, they end up in a state where their policies conflict (at the static policy level).

Example 5. *Imagine that Alice and Bob want to leverage the advantages of evolving policies, and they rewrite the previous static policies in a more precise way,* "Everyone can see the posts on my timeline during my birthday" *and* "Only my friends can see my posts when I am at home". *Combining the policy automata representing these two policies, we can identify a conflict in a state reachable after a trace in which, Alice's birthday begins and afterwards (before the day ends) Bob goes home. Note that it is not required that Bob posts in Alice's timeline for the conflicting policies to be reached, since it is known beforehand that both policies cannot be satisfied at the same time.*

Based on this definition and the assumptions we made about conflicts over static policies, we can prove that evolving policies are closed under increasing strictness.

Theorem 1. *Given the policy automata $\mathcal{P}_1$ and $\mathcal{P}_2$ the following holds*

$$\mathcal{P}_1 \maltese_{PA} \mathcal{P}_2 \;\; \wedge \;\; \mathcal{P}_1' \sqsubseteq_{PA} \mathcal{P}_1 \implies \mathcal{P}_1' \maltese_{PA} \mathcal{P}_2.$$

V.3 Translation of policy automata to DATEs

Dynamic Automata with Timers and Events (DATEs) [17] are symbolic automata aimed at representing monitors, with a corresponding compilation tool LARVA. In this section, we introduce the basic definitions (leaving out advanced element which are not necessary for this paper) enabling us to provide the translation from policy automata, effectively providing an implementation to the latter through LARVA. As a monitoring formalism, DATE transitions are *event, condition, action* triples: if a matching event occurs and the condition — based on event parameters and the automaton symbolic state — holds, then the action is carried out. The action can be used to either modify the automaton state, interact with the event-generating system, or generate an alert as appropriate.

Definition 8. *A* symbolic automaton *(*SA*) running over a system with state of type Θ, is a quintuple $\langle Q,\ q_0,\ a_0,\ \rightarrow,\ B\rangle$ with set of states Q, initial state $q_0 \in Q$, initial action to be executed $a_0 \in \Theta \to \Theta$, transition relation $\rightarrow \subseteq Q \times \text{event} \times (\Theta \to \mathbb{B}) \times (\Theta \to \Theta) \times Q$, and bad states $B \subseteq Q$. Note that the transitions between automaton states are labelled with: (i) an event expression which triggers the transition; (ii) an enabling condition on the system state — encoded as a function from the system state to a boolean value; and (iii) an action (code) which may change the state of the underlying system — encoded as a function, which given a system state returns an updated system state.*

A total ordering $<$, giving a priority to transitions, is assumed to be given so as to ensure determinism.

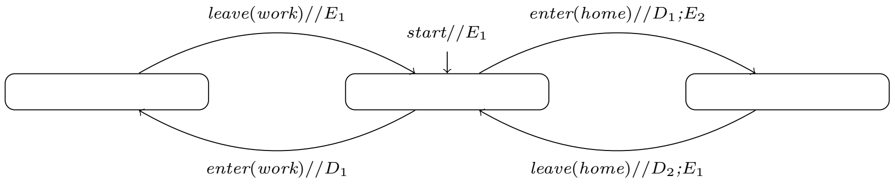
The behaviour of an SA M, upon receiving a set of events, consists of: (i) choosing the enabled transition with the highest priority; (ii) performing the transition (possibly triggering a new set of events); and (iii) repeating until no further events are generated, upon which the automaton waits for a system event.

V.3.1 Translation

Intuitively, the translation keeps the same states of the policy automaton, but introduces transitions and states for each static policy. We note that the translation below only handles the high-level enabling and disabling of policies, leaving the low-level checking and enforcement up to a static policy checker. We note that the translation below only handles conjunction of policies.

Given a policy automaton $\langle \Sigma, Q, q_0, \rightarrow, \pi \rangle$, for a given transition $(q, e, q') \in \rightarrow$, we generate an action which disables policies in the outgoing state, and enabling those in the ingoing state, as follows: $\text{action}(q, e, q') = \mathit{stopEnforcing}(\pi(q)); \mathit{startEnforcing}(\pi(q'))$, where *startEnforcing*$(p)$ and *stopEnforcing*(p) switches on and off the enforcement of static policy p. Using this construction, we generate transitions of the SA labelled as follows: $\rightarrow_{\text{SA}} = \{(q, e, \text{true}, \text{action}(q, e, q'), q') \mid (q, e, q') \in \rightarrow\}$. The resulting DATE would be: $\langle Q, q_0, \text{start}, \rightarrow_{\text{SA}}, \emptyset \rangle$ where start is an action representing the activation of the automaton.

Example 6. *Consider the policy automata presented in Example 1, which models the policy* 'Co-workers cannot see my posts while I am not at work, and only family can see my location while I am at home'*. Assuming that the events leave*(*work*)*, leave*(*home*)*, enter*(*work*) *and enter*(*home*) *exist, the automaton can be directly converted to a DATE as follows:*



where E_1, D_1, E_2 and D_2 are defined as follows:

$$\begin{aligned}
E_1 &= \mathit{startEnforcing}(\mathcal{F}_{\mathit{co\text{-}workers}}(\mathit{read\text{-}post}))\\
D_1 &= \mathit{stopEnforcing}(\mathcal{F}_{\mathit{co\text{-}workers}}(\mathit{read\text{-}post}))\\
E_2 &= \mathit{startEnforcing}(\mathcal{F}_{\mathit{co\text{-}workers}}(\mathit{read\text{-}post}) \;\&\; \mathcal{F}_{\overline{\mathit{family}}}(\mathit{see\text{-}location}))\\
D_2 &= \mathit{stopEnforcing}(\mathcal{F}_{\mathit{co\text{-}workers}}(\mathit{read\text{-}post}) \;\&\; \mathcal{F}_{\overline{\mathit{family}}}(\mathit{see\text{-}location}))
\end{aligned}$$

V.4 Implementation in Diaspora* using LARVA

One of our objectives is to have an effective enforcement mechanism for evolving privacy policies based on policy automata in a real OSN. In this section, we describe the details of the implementation of policy automata using LARVA in the OSN Diaspora*.

We chose Diaspora* since it is open source, which allows us to implement the interaction between the OSN and LARVA. Diaspora* has a built-in mechanism for enforcing static privacy policies. Pardo and Schneider have recently extended Diaspora* with a prototype implementation of some privacy policies defined in the $\mathcal{PPF}$ framework [71, 67]. $\mathcal{PPF}$ is a formal (generic) privacy policy framework for OSNs, which needs to be instantiated for each OSN in order to take into account the specificities of the OSN. $\mathcal{PPF}$ was shown not only to be able to capture all privacy policies of Twitter

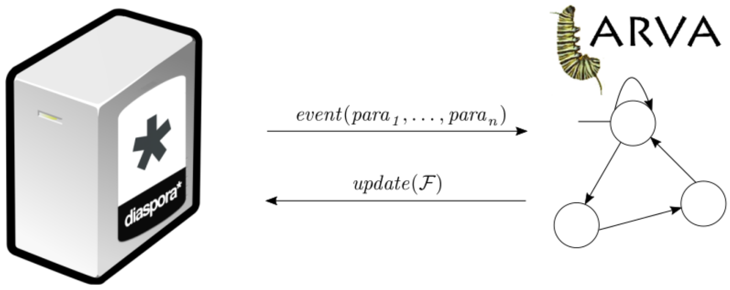


Figure V.1: High-level representation of the Diaspora*-LARVA communication

and Facebook, but also more complex ones involving implicit disclosure of information. $\mathcal{PPF}$ comes with a privacy policy language, $\mathcal{PPL}$, which satisfies all the assumptions placed for the static privacy language in policy automata (cf. Section V.2).

Using policy automata to model the evolution of the privacy policies makes it possible to define a modular enforcement of evolving policies. As we mentioned, policy automata are independent of the static policy language of the OSN (except for the assumptions on $=_{SPL}$ and &), and consequently, they are also independent of the underlying enforcement of each particular static policy. Policy automata can be translated to DATEs (cf. Section V.3). In order to implement policy automata we use the tool LARVA [18], which automatically generates a monitor from properties expressed in DATEs.

In order for the runtime enforcement to work we use a communication protocol between Diaspora* and LARVA. Every time a relevant event occurs in Diaspora* (i.e., an event that can update the state of the automata), it is reported to LARVA. Then LARVA updates the state of the privacy policies (if applicable), and whenever a privacy policy is updated LARVA reports this change to Diaspora*, which would update the corresponding (static) privacy policy (see Fig. V.1).

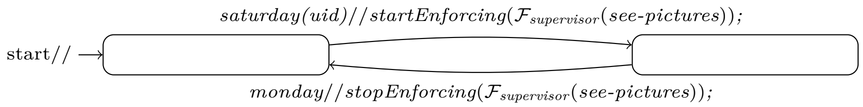
Given that Diaspora* is implemented in Ruby and the monitors that LARVA generates are Java programs, we implement the communication protocol using sockets. One socket is used by Diaspora* to send a message to LARVA containing the event that has occurred, plus additional information such as the user who triggered the event; if it is a post the audience of the post, whether the post contains a location, etc. LARVA monitors detect (among other things) Java method calls corresponding to events on DATE transitions. Therefore, we have implemented a Java program, which listens to the communication socket and depending on the message sent by Diaspora* it calls a concrete method causing the LARVA automaton to update its state. When an automaton updates its state, the privacy policies to be enforced might change. There is another socket that the LARVA monitor uses to send the privacy policies that Diaspora* should enforce. The message sent by the monitor includes the policies that must be activated (policies of the incoming state) and/or deactivated (policies of the outgoing state). This part of the communication will also be handled by the Java program, which contains an auxiliary method for sending messages to Diaspora*.

V.5 Case studies

As a proof-of-concept we have implemented two policy automata in the Disapora*-LARVA system presented in the previous section. Here we describe the concrete details of this prototype. The code of these case studies can be found in [21].

V.5.1 Case 1: Protecting pictures during the weekend

In this case study we describe the implementation of the following evolving privacy policy, *"My supervisor cannot see my pictures during the weekend"*. This is a simple policy that only depends on the time of the week. Let $\mathcal{F}_{supervisor}(\textit{see-pictures})$ represent that my supervisor cannot see my pictures, the following DATE models the policy


$saturday(uid)//startEnforcing(\mathcal{F}_{supervisor}(see\text{-}pictures));$
start//
$monday//stopEnforcing(\mathcal{F}_{supervisor}(see\text{-}pictures));$

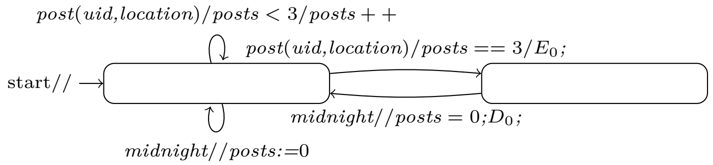
As we mentioned, Diaspora*'s privacy protection mechanism is based on an instantiation of $\mathcal{PPF}$. In this instantiation, we consider that a user appears in a picture if the user is mentioned in the post containing the picture[5]. For this policy automaton Diaspora* is required to report the events *saturday* and *monday*. Each of them represents the beginning of the day after which they are named. Every Saturday at 00:00 Diaspora* sends the message `uid;saturday` to LARVA where `uid` is a user id. This message is sent once for each user with her corresponding `uid`. At this point the automaton of each user is updated. The automaton moves to the only possible state where it replies with the message `uid;exclude-supervisor;picture`. When this message is received by Diaspora*, it activates the static privacy policy that forbids posting a picture of a user if her supervisor is part of the audience. More precisely, Diaspora*'s built-in enforcement mechanism will block any post that contains a picture and mention of a user whose supervisor is included in the audience of the post. Similarly, on Monday at 00:00, Diaspora* informs the automata with the message `monday`. All active automata update their state, therefore no `uid` parameter is needed for this event. This choice also reduces

[5]Diaspora* does not support tagging users in pictures.

the amount of messages sent between Diaspora* and LARVA. Finally, these automata reply to Diaspora* with the message `uid;include-supervisor;picture`, which allows again the user's supervisor to be part of the audience of her pictures.

V.5.2 Case 2: Disclosing location at most 3 times per day

Here we describe the implementation of the policy automaton of Example 2, which we translate to a DATE (as described in Section V.3) as follows


post(*uid,location*)/*posts* < 3/*posts* + +
post(*uid,location*)/*posts* == 3/*E*₀;
start//
midnight//*posts* = 0;*D*₀;
midnight//*posts*:=0

In the previous automaton $E_0 = \mathit{startEnforcing}(\mathcal{F}_{all}(\mathit{post}(\mathit{uid,location})))$ and $D_0 = \mathit{stopEnforcing}(\mathcal{F}_{all}(\mathit{post}(\mathit{uid,location})))$. Note that we use the variable *posts* to symbolically encode the explict states of the real policy automata (cf. Section V.2). There are two events present in the transitions of the automaton, which therefore need to be reported from Diaspora* to the LARVA monitors when they occur, *post*(*uid,location*) and *midnight*.

In our Diaspora* $\mathcal{PPF}$ instantiation, mentioning users in a post that includes a location constitutes a disclosure of their location. Every time a user is mentioned in a post (i.e., *post*(*uid,location*)), a message including the message `uid;post;location` is sent to LARVA, specifying the user id and that a location of this user has been disclosed. The message is sent for each user mentioned in the post. As described before, there is one LARVA monitor per user, which controls the policy automaton of each individual. When the message is received the automaton of the user specified by `uid` will be updated. This update will increase the value of the automaton variable *posts*, whose initial value is 0. After sending the message, Diaspora* waits for the answer of the automaton, in case an update of the privacy policies of the user is required. In case *posts* is less than 3, there is no need to update the privacy policies, therefore the message `do-nothing` is sent back. On the other hand, if *posts* is greater than 3, the automaton will move to the state where the policy forbidding the disclosure of locations must be activated, thus it will send the message `disable-posting` to Diaspora*. Note that it is not required to specify the user id in the reply since Diaspora* initiated the communication.

As for the event *midnight*, Diaspora* sends the message `midnight` to the monitors of all users every day at 23:59. If the monitors are in the state where the disclosure of location is forbidden, they take the transition to the initial state. This transition involves, firstly, resetting the variable *posts* to 0, and secondly, sending the message `uid;enable-posting;location` back to Diaspora*, which removes the privacy policy preventing the location of the user `uid` to be disclosed. If the automaton is already in the initial state, it simply resets *posts* to 0.

V.6 Related work

The lack of a temporal dimension in privacy policies was already pointed out by Riesner *et al.* [78]. In their survey, they show that there is no OSN that supports policies that automatically change over time. The authors mention that Facebook allows users to apply a default audience to all their own old posts, but there is a big gap between that privacy policy and the family of evolving policies that we introduce in this paper.

Specifying and reasoning about temporal properties in multi-agent systems using epistemic logic have been the subject of study for a long time. It began with the so called *interpreted systems* (IS). In [29] Fagin *et al.* introduce IS as a model to interpret epistemic formulae with temporal operators such as box and diamond. IS have been used for security analyses of multi-agent systems. Though we do consider a temporal aspect, the focus and objectives of our work are different from the work done in interpreted systems, at least in what concerns the domain of application and the scope of the approach. In our case, the policies themselves are the ones evolving based on events, rather than the information on what is known to different agents at a given time.

Recent research has been carried out in extending IS to be able to reason about past or future knowledge. In [8] Ben-Zvi and Moses extend K_i with a timestamp $K_{i,t}$, making it possible to express properties such as "Alice knows at time 5 that Bob knew p at time 3", i.e., $K_{Alice,5}K_{Bob,3}\ p$. With the same essence but including real time, Woźna and Lomuscio present TCTLKD [97], a combination of epistemic logic, CTL, a deontic modality and real time. In these, and other related work, the intention is to be able to model the time differences in the knowledge acquired by different agents due to delay in communication channels. Although both our motivation as well as the application domain differ from those of the aforementioned logics, they could be indeed useful to express certain real-time policies not currently supported in our formalism.

Despite the richness of both timed epistemic logics, TCTLKD [97] and the epistemic logic with timestamps [8], they would not be able to express recurrent policies as we do. We are of course adding a separate layer beyond the power of the logical formalism by using automata to precisely express when to switch from one policy to another. It remains an interesting question what would be the expressivity of policy automata if we consider an enhancement of $\mathcal{PPF}$ with timed extensions as done in some of the above works in order to express richer (static) policies.

We have not defined here a theory of privacy policies (we have not given a formal definition in terms of traces or predicates), nor have we developed a formal theory of enforcement of privacy policies. To the best of our knowledge such a characterisation does not exist for privacy policies. There is, however, work done in the context of security policies, for instance the work by Le Guernic *et al.* on using automata to monitor and enforce non-interference [43, 37] or by Schneider on security automata [83]. It could be instructive to further develop the theoretical foundations of policy automata and relate it to security automata and their successors (e.g., edit automata [49]).

V.7 Conclusions

We have presented a novel technique to define and implement evolving privacy policies (i.e., recurrent policies that are (de)activated depending on events) for OSNs. We have defined policy automata as a formalism to express about such policies. Moreover, we have introduced the notion of parallel composition, subsumption and conflict between policy automata and we have proved some of their properties. We have defined a translation from policy automata to DATEs which enables their implementation by means of the tool LARVA. Furthermore, we have describe how to connect LARVA monitors to the OSN Diaspora* so that policy automata can effectively be implemented. In fact, the presented approach would allow to plug in policy automata to any OSN with a built-in enforcement of static privacy policies. Finally, as a proof-of-concept, we have implemented a prototype of two evolving privacy policies.

The policy automata approach has some limitations. For instance, consider that Alice enables the following policy *"Only my friends can see my pictures during the weekend"*. Imagine that Alice and Bob are not friends. If Alice shares a picture on Saturday, Bob will not have access to it. However, on Monday this policy would be deactivated. What would be the effect of turning off this policy? It might be possible that Bob gains access to all the pictures that Alice posted during the weekend, since no restrictions are specified outside the scope of the weekend. In order to address this problem we might need a policy language able to express *real-time* aspects, with an element of access memory integrated within policy automata.

We are currently also extending policy automata with timing events such as timeouts. This extension will be almost immediately implementable using LARVA since DATEs already support timeouts in their transitions. Another line of work is to extend policy automata with location events. Users normally access OSNs through mobile devices. These devices could directly report the location of users to their policy automata, which avoids having to constantly report users' location to the OSN.

V.A Appendix

V.A.1 Proofs

Lemma 1. *The relation $\sqsubseteq_{PA}$ is transitive, antisymmetric and reflexive.*

Proof. We proof each property separately:

- $\underline{\sqsubseteq_{PA} \text{ is reflexive}}$ We show that for all policy automata $\mathcal{P}$ the following holds

  $$\mathcal{P} \sqsubseteq_{PA} \mathcal{P}$$

  which by the definition of $\sqsubseteq_{PA}$ is equivalent to

  $$\mathcal{P} \|_{\Sigma} \mathcal{P} =_{PA} \mathcal{P}$$

  and by the definition of $=_{PA}$, it is equal to

  $$\forall es \in \Sigma^* \cdot policy_{\mathcal{P}\|\mathcal{P}}(es) =_{SPL} policy_{\mathcal{P}}(es)$$

  We show it by induction on the length of the trace.

  Base case. Let ε be the empty trace. We show that the following holds

  $$policy_{\mathcal{P}\|\mathcal{P}}(\varepsilon) =_{SPL} policy_{\mathcal{P}}(\varepsilon)$$

  Given that the trace is empty both $\mathcal{P}$ is in its initial state q_0 and by the definition of *policy*() (cf. Def. 2) the following holds

  $$\pi(q_0) \& \pi(q_0) =_{SPL} \pi(q_0)$$

  By Assumption 2, & is an idempotent relation therefore

  $$\pi(q_0) =_{SPL} \pi(q_0)$$

  Inductive step. For all $es \in \Sigma^*$ and $e \in \Sigma$. We assume that

  $$policy_{\mathcal{P}\|\mathcal{P}}(es) =_{SPL} policy_{\mathcal{P}}(es)$$

  holds (Inductive hypothesis) and we show that the following holds

  $$policy_{\mathcal{P}\|\mathcal{P}}(e : es) =_{SPL} policy_{\mathcal{P}}(e : es)$$

Let q be the state of $\mathcal{P}$ after the execution of $e : es$, by the definition of *policy*() the following holds

$$\pi(q)\&\pi(q) =_{SPL} \pi(q)$$

Since & is an idempotent relation it holds that

$$\pi(q) =_{SPL} \pi(q)$$

After the execution of e we have that

$$policy_{\mathcal{P}\|\mathcal{P}}(es) =_{SPL} policy_{\mathcal{P}}(es)$$

which holds by inductive hypothesis.

- $\sqsubseteq_{PA}$ is transitive We show that for all policy automata $\mathcal{P}_1$, $\mathcal{P}_2$ and $\mathcal{P}_3$ the following holds

$$\text{If } \mathcal{P}_1 \sqsubseteq_{PA} \mathcal{P}_2 \text{ and } \mathcal{P}_2 \sqsubseteq_{PA} \mathcal{P}_3 \text{ then } \mathcal{P}_1 \sqsubseteq_{PA} \mathcal{P}_3$$

As before, from the definition of $\sqsubseteq_{PA}$ and $=_{PA}$ (cf. Def. 6 and Def. 5, respectively) we can rewrite the previous statement as

$$\forall es \in \Sigma^* \cdot policy_{\mathcal{P}_1\|\mathcal{P}_2}(es) =_{SPL} policy_{\mathcal{P}_1}(es) \tag{V.1}$$

and

$$\forall es \in \Sigma^* \cdot policy_{\mathcal{P}_2\|\mathcal{P}_3}(es) =_{SPL} policy_{\mathcal{P}_2}(es) \tag{V.2}$$

then

$$\forall es \in \Sigma^* \cdot policy_{\mathcal{P}_1\|\mathcal{P}_3}(es) =_{SPL} policy_{\mathcal{P}_1}(es)$$

We show it by induction on the length of the trace.

Base case. Let ε be the empty trace. We show that the following holds

$$policy_{\mathcal{P}_1\|\mathcal{P}_3}(\varepsilon) =_{SPL} policy_{\mathcal{P}_3}(\varepsilon)$$

Let q_0^1, q_0^2 and q_0^3 be the initial states of the policy automata $\mathcal{P}_1$, $\mathcal{P}_2$ and $\mathcal{P}_3$, respectively. We rewrite the premises (V.8) and (V.9) above as

$$\pi(q_0^1)\&\pi(q_0^2) =_{SPL} \pi(q_0^1) \tag{V.3}$$

$$\pi(q_0^2)\&\pi(q_0^3) =_{SPL} \pi(q_0^2) \tag{V.4}$$

and we want to show that

$$\pi(q_0^1)\&\pi(q_0^3) =_{SPL} \pi(q_0^1) \tag{V.5}$$

By Equation V.3, the previous goal can be rewritten as

$$\pi(q_0^1)\&\pi(q_0^2)\&\pi(q_0^3) =_{SPL} \pi(q_0^1)$$

which trivially follows by replacing $\pi(q_0^2)$ on the left hand of (V.3) with $\pi(q_0^2)\&\pi(q_0^3)$ (cf. equation (V.4)).

Inductive step. For all $es \in \Sigma^*$ and $e \in \Sigma$. We assume that

$$policy_{\mathcal{P}_1\|\mathcal{P}_3}(es) =_{SPL} policy_{\mathcal{P}_1}(es)$$

holds (Inductive hypothesis) and we show that the following holds

$$policy_{\mathcal{P}_1\|\mathcal{P}_3}(e : es) =_{SPL} policy_{\mathcal{P}_1}(e : es)$$

Let q_1, q_2 and q_3 be states of the policy automata $\mathcal{P}_1$, $\mathcal{P}_2$ and $\mathcal{P}_3$ after the execution of $e : es$, respectively. We rewrite the premises above as

$$\pi(q_1)\&\pi(q_2) =_{SPL} \pi(q_1)$$

$$\pi(q_2)\&\pi(q_3) =_{SPL} \pi(q_2)$$

and we want to show that

$$\pi(q_1)\&\pi(q_3) =_{SPL} \pi(q_1)$$

which follows by the exact same reasoning as in the base case. After the execution of e we have that

$$policy_{\mathcal{P}_1\|\mathcal{P}_3}(es) =_{SPL} policy_{\mathcal{P}_3}(es)$$

which holds by inductive hypothesis.

- $\underline{\sqsubseteq_{PA} \text{ is antisymmetric}}$ We show that for all policy automata $\mathcal{P}_1$ and $\mathcal{P}_2$ the following holds

$$\text{If } \mathcal{P}_1 \sqsubseteq_{PA} \mathcal{P}_2 \text{ and } \mathcal{P}_2 \sqsubseteq_{PA} \mathcal{P}_1 \text{ then } \mathcal{P}_1 =_{PA} \mathcal{P}_2 \tag{V.6}$$

$$\text{If } \mathcal{P}_1 \neq_{PA} \mathcal{P}_2 \text{ and } \mathcal{P}_1 \sqsubseteq_{PA} \mathcal{P}_2 \text{ then } \mathcal{P}_2 \not\sqsubseteq_{PA} \mathcal{P}_1 \tag{V.7}$$

Statement (V.6) follows from reflexivity of $\sqsubseteq_{PA}$.

As before, by the definition of $\sqsubseteq_{PA}$ and $=_{PA}$ Statement (V.7) can be rewritten as

$$\exists es' \in \Sigma^* policy_{\mathcal{P}_1}(es') \neq_{SPL} policy_{\mathcal{P}_2}(es') \tag{V.8}$$

and

$$\forall es \in \Sigma^* \cdot policy_{\mathcal{P}_1 \| \mathcal{P}_2}(es) =_{SPL} policy_{\mathcal{P}_1}(es) \tag{V.9}$$

then

$$\exists es'' \in \Sigma^* \cdot policy_{\mathcal{P}_2 \| \mathcal{P}_1}(es'') \neq_{SPL} policy_{\mathcal{P}_2}(es'')$$

We show it by induction on the length of the trace.

Base case. Let ε be the empty trace. We show that if

$$policy_{\mathcal{P}_1}(\varepsilon) \neq_{SPL} policy_{\mathcal{P}_2}(\varepsilon)$$

and

$$policy_{\mathcal{P}_1 \| \mathcal{P}_2}(\varepsilon) =_{SPL} policy_{\mathcal{P}_1}(\varepsilon)$$

then

$$policy_{\mathcal{P}_2 \| \mathcal{P}_1}(\varepsilon) \neq_{SPL} policy_{\mathcal{P}_2}(\varepsilon)$$

Let q_0^1 and q_0^2 be the initial states of the policy automata $\mathcal{P}_1$ and $\mathcal{P}_2$, respectively. We rewrite the premises above as

$$\pi(q_0^1) \neq_{SPL} \pi(q_0^2)$$

$$\pi(q_0^1) \& \pi(q_0^2) =_{SPL} \pi(q_0^1)$$

By replacing $\pi(q_0^1)$ on the first conjunct we get

$$\pi(q_0^1) \& \pi(q_0^2) \neq_{SPL} \pi(q_0^2)$$

given that both automata are in the initial state, it is equivalent to

$$policy_{\mathcal{P}_2 \| \mathcal{P}_1}(\varepsilon) \neq_{SPL} policy_{\mathcal{P}_2}(\varepsilon)$$

as required.

Inductive step. Let $es \in \Sigma^*$ and $e \in \Sigma$. We assume that

$$policy_{\mathcal{P}_2 \| \mathcal{P}_1}(es) \neq_{SPL} policy_{\mathcal{P}_2}(es)$$

holds (Inductive hypothesis) and we show that the following holds

$$policy_{\mathcal{P}_2 \| \mathcal{P}_1}(e : es) \neq_{SPL} policy_{\mathcal{P}_2}(e : es)$$

Let q_1 and q_2 be states of the policy automata $\mathcal{P}_1$ and $\mathcal{P}_2$ after the execution of $e : es$, respectively. We rewrite the premises above as

$$\pi(q_1) \neq_{SPL} \pi(q_2)$$

$$\pi(q_1)\&\pi(q_2) =_{SPL} \pi(q_1)$$

and we want to show that

$$\pi(q_2)\&\pi(q_1) \neq_{SPL} \pi(q_2)$$

which follows by the exact same reasoning as in the base case. After the execution of e we have that

$$policy_{\mathcal{P}_2 \| \mathcal{P}_1}(es) \neq_{SPL} policy_{\mathcal{P}_2}(es)$$

which holds by inductive hypothesis.

□

□

Theorem 1. *Given the policy automata $\mathcal{P}_1$ and $\mathcal{P}_2$ the following holds*

$$\mathcal{P}_1 \maltese_{PA} \mathcal{P}_2 \quad \wedge \quad \mathcal{P}_1' \sqsubseteq_{PA} \mathcal{P}_1 \implies \mathcal{P}_1' \maltese_{PA} \mathcal{P}_2$$

Proof. From the assumption the following holds

$$\mathcal{P}_1 \maltese_{PA} \mathcal{P}_2 \tag{V.10}$$

By the definition of $\maltese_{PA}$ (cf. Def. 7) the following holds

$$\exists trc \in \Sigma^* \cdot policy_{\mathcal{P}_1}(trc) \maltese_{SPL} policy_{\mathcal{P}_2}(trc)$$

Let q_1 and q_2 be the states of $\mathcal{P}_1$ and $\mathcal{P}_2$ after executing *trc* (respectively), then by the definition of *policy*() (cf. Def. 2) the following holds

$$\pi(q_1) \maltese_{SPL} \pi(q_2)$$

We also assume that

$$\mathcal{P}_1' \sqsubseteq_{PA} \mathcal{P}_1$$

By the definition of $\sqsubseteq_{PA}$ (cf. Def. 6) we have that

$$\forall tr \in \Sigma^* \cdot policy_{\mathcal{P}_1 \| \mathcal{P}_1'}(tr) =_{SPL} policy_{\mathcal{P}_1'}(tr)$$

Since $trc \in \Sigma^*$ it also holds that

$$policy_{\mathcal{P}_1 \| \mathcal{P}_1'}(trc) =_{SPL} policy_{\mathcal{P}_1'}(trc)$$

Let q_1 and q_1' be the states of $\mathcal{P}_1$ and $\mathcal{P}_1$' after executing *trc* (respectively), then by the definition of *policy*() it holds that

$$\pi(q_1)\&\pi(q_1) =_{SPL} \pi(q_1') \tag{V.11}$$

Since we use the same trace *trc*, by Equation (V.10) and Assumption 3 it follows that

$$\pi(q_1)\&\pi(q_1')\maltese_{SPL}\pi(q_2)$$

By Equation (V.11) we can replace $\pi(q_1)\&\pi(q_1')$

$$\pi(q_1')\maltese_{SPL}\pi(q_2)$$

Finally, by the definition of $\maltese_{PA}$ we conclude that

$$\mathcal{P}_1'\maltese_{SPL}\mathcal{P}_2$$

$\square$

Chapter VI

Specification of Evolving Privacy Policies for Online Social Networks

Raúl Pardo, Ivana Kellyérová, César Sánchez and Gerardo Schneider

Abstract. *Online Social Networks* are ubiquitous, bringing not only numerous new possibilities but also big threats and challenges. Privacy is one of them. Most social networks today offer a limited set of (static) privacy settings, not being able to express *dynamic* policies. For instance, users might decide to protect their location during the night, or share information with difference audiences depending on their current position. In this paper we introduce $\mathcal{PPF}_{\mathcal{T}}$, a formal framework to express, and reason about, *dynamic* (and *recurrent*) privacy policies that are activated or deactivated by context (events) or time. Besides a formal policy language ($\mathcal{PPL}_{\mathcal{T}}$), the framework includes a knowledge-based logic extended with (linear) temporal operators and a *learning* modality ($\mathcal{KBL}_{\mathcal{T}}$). Policies, and formulae in the logic, are interpreted over (timed) traces representing the evolution of the social network. We prove that checking privacy policy conformance, and the model-checking problem for $\mathcal{KBL}_{\mathcal{T}}$, are both decidable.

VI.1 Introduction

Online Social Networks, also known as *Social Networking Sites* or *Social Network Services*, have exploded in popularity in the last years. Over the past decade, the use of Facebook [28] and Twitter [94], just to mention two of the most popular ones, has increased at the point of becoming ubiquitous. Nearly 70% of the Internet users are active on social networks as shown by a recent survey [45], and this number is increasing. A number of studies show that the number of privacy breaches is keeping pace with this growth [56, 41, 51, 57]. Very often users' requirements are far from the privacy guarantees offered by social networks which do not meet their expectations. The reasons for that are multifold, ranging from the users' lack of knowledge on the underlying technology to fundamental technical issues of the technology itself.

We are here only concerned with the fact that the privacy settings currently available in social networks are not suitable for capturing the *dynamic* aspect of privacy policies. That is, privacy policies should take into account that the networks *evolve*, as well as the privacy preferences of the users. The privacy policy may "evolve" due to explicit changes done by the users (e.g., a user may change the audience of an intended post to make it more restrictive), or because the privacy policy is dynamic *per se*. Examples of the latter, are for instance: *"My boss cannot know my location between 20:00-23:59 every day"*, or *"Only my friends can see the pictures I am tagged in from Fridays at 20:00 till Mondays at 08:00"*. These are recurrent policies triggered by some time events ("every day between 20:00 and 23:59", and "every week from Friday at 20:00 till Monday at 08:00"). Other policies may be activated or deactivated by certain events: *"Only up to 3 posts, disclosing my location, are allowed per day in my timeline"*.

In this paper we present a formal framework to express evolving privacy policies. We take $\mathcal{FPPF}$ [71] as a point of departure. $\mathcal{FPPF}$ is a formal framework for privacy policies which consists of: i) A generic social network model (SNM); ii) A knowledge-based logic ($\mathcal{KBL}$) to reason about the social network and privacy policies; iii) A formal language ($\mathcal{PPL}$) to describe privacy policies (based on the previous logic). $\mathcal{FPPF}$ is a an expressive privacy policy framework able to represent all privacy policies for social networks like Facebook and Twitter, and beyond [71]. Though rich in what concerns its expressiveness, $\mathcal{FPPF}$ is not suitable to express evolving privacy policies, in the sense discussed above. In summary, our contributions in this paper are:

i) We introduce $\mathcal{PPF}_{\mathcal{T}}$, an extension of $\mathcal{FPPF}$ able to represent recurrent privacy policies parametrised with timed intervals. For that we syntactically extend the privacy policy language $\mathcal{PPL}$ with timed intervals and recurrent behaviour (*timed* $\mathcal{PPL}$), and the underlying epistemic logic $\mathcal{KBL}$ with temporal operators (*box* and *diamond*) and a *learn* operator (*temporal* $\mathcal{KBL}$). Policies in $\mathcal{PPL}$, and formulae in

$\mathcal{KBL}$, are interpreted over (finite) timed traces.

ii) We study properties of both the new privacy policy language and the underlying logic, in particular showing that the new operator *learn* cannot be derived. We prove decidability of the satisfaction relation for the logic, and of the conformance relation for the policy language.

The paper is structured as follows, Section VI.2 introduces all the elements of the new privacy policy framework. More precisely, in Section VI.2.1 we introduce temporal $\mathcal{KBL}$; in Section VI.2.2 we describe timed SNMs, we define a satisfaction relation for formulae of the previous logic, we show that the model-checking problem is decidable, and study the properties of the new learning modality; in Section VI.2.3 we present the privacy policy language timed $\mathcal{PPL}$, we define what means for a privacy policy to be in conformance to a timed SNM and show that this procedure is decidable. In Section VI.3 we analyse the complexity of the $\mathcal{KBL}$ model checking problem, and we provide an alternative optimised model checking algorithm. All the proofs are in the appendix with the exception of Theorem 1 which is presented in Section VI.3.

VI.2 Timed $\mathcal{FPPF}$

In this section, we introduce the extension of $\mathcal{FPPF}$ with time, which contains the following elements:

a) A knowledge-based logic $\mathcal{KBL}_\mathcal{T}$ with additional temporal modalities, inspired by temporal logics such as LTL.

b) A social network model (as defined for $\mathcal{FPPF}$), together with the notion of traces, i.e., sequences of these models that capture the evolution of a social network which we use to give semantics to the previous logic. We use $\mathcal{SN}$ to denote the universe of all possible social network models.

c) A privacy policy language ($\mathcal{PPL}_\mathcal{T}$), enabling the user to define a (possibly recurring) time window in which their policy should be enforced.

Together, these parts form the new *Timed First-Order Privacy Policy Framework*, $\mathcal{PPF}_\mathcal{T}$. In the following sections, we describe each of the components separately.

VI.2.1 Temporal $\mathcal{KBL}$

$\mathcal{KBL}_\mathcal{T}$ is a *temporal knowledge-based logic* for social networks. It contains the knowledge modality present in all epistemic logics [29] and the temporal modalities box and diamond. Additionally, it includes: i) Two special types of predicates, *connection* and *action* predicates. Connection predicates represent the "social" connections between users. For instance, *friends*, *colleagues*, *family*, *co-workers*, and so forth. On the other hand, action predicates model the actions which are permitted to be executed by a user. For example, Alice can send a friend request to Bob or Alice can join events created by Bob. We use $\mathcal{C}$ and Σ to denote the set of connection and action predicates, respectively; ii) A modality to represent learning. It will allow us to differentiate between the moment some piece of information has been learnt or whether it is known. We study the relation between the learning and knowledge modalities in the following section.

Let $\mathcal{T}$ be a set (which will refer as *vocabulary*) which consists of *predicate symbols* (p), *function symbols* (f) and *constant symbols* (c). Predicate and function symbols have some implicit arity which corresponds to the number of arguments they take. We assume an infinite supply of variables, which we write as x, y and so on. We can form terms using the elements of $\mathcal{T}$ as follows: $s ::= c \mid x \mid f_i(\vec{s})$ where $\vec{s}$ is a tuple of terms respecting the arity of f_i. Let Ag be a finite set of *agents*. The syntax of $\mathcal{KBL}_{\mathcal{T}}$ is defined as follows.

Definition 1 (Syntax of $\mathcal{KBL}_{\mathcal{T}}$)**.** *Given agents $i, j \in Ag$, a nonempty set of agents $G \subseteq Ag$, the predicate symbols $a_n(i,j)$, $c_m(i,j)$, $p(\vec{s})$ where $m \in \mathcal{C}$ and $n \in \Sigma$, and a variable x. The* syntax of the timed knowledge-based logic $\mathcal{KBL}_{\mathcal{T}}$ *is inductively defined as:*

$$\begin{array}{lcl} \varphi & ::= & \varphi \wedge \varphi \mid \neg\varphi \mid \forall x.\varphi \mid \Box\varphi \mid \Diamond\varphi \mid \xi \\ \xi & ::= & \psi \mid L_i\psi \\ \psi & ::= & \rho \mid \psi \wedge \psi \mid \neg\psi \mid \forall x.\psi \mid K_i\psi \mid D_G\psi \\ \rho & ::= & c_m(i,j) \mid a_n(i,j) \mid p(\vec{s}) \end{array}$$

The remaining epistemic modalities are defined as follows: $S_G\varphi \triangleq \bigvee_{i\in G} K_i\varphi$; $E_G\varphi \triangleq \bigwedge_{i\in G} K_i\varphi$; $SL_G\varphi \triangleq \bigvee_{i\in G} L_i\varphi$; $EL_G\varphi \triangleq \bigwedge_{i\in G} L_i\varphi$. □

Note that in the version of the logic presented here L_i cannot appear within the scope of a K_j operator. We use $\mathcal{F}_{\mathcal{TKBL}}$ to denote the set of all well-formed formulae of $\mathcal{KBL}_{\mathcal{T}}$ according to category φ. Formulae in ψ, represent the $\mathcal{KBL}$ logic [67]; $\mathcal{F}_{\mathcal{KBL}}$ denote the set of well-formed $\mathcal{KBL}$ formulae.[1] The epistemic modalities stand for: $K_i\varphi$, agent i knows φ; $L_i\varphi$, agent i learnt φ; $S_G\varphi$, someone in the group G knows φ; $E_G\varphi$, everyone in the group G knows φ; $SL_G\varphi$, someone in the group G learnt φ; $EL_G\varphi$, everyone in the group G learnt φ; $D_G\varphi$, φ is distributed knowledge in the group G. We use the following operators as syntactic sugar for permission: $P_i^j a_n := a_n(i,j)$, agent i is permitted to execute action a_n to agent j; $SP_G^j a_n :=$

[1]For simplicity of presentation we leave out the *common knowledge* operator from the logic. Though rather technical its treatment would only need the machinery from standard epistemic logic.

$\bigvee_{i \in G} a_n(i,j)$, at least one agent in G is permitted to execute action a to agent j; $GP^j_G a_n := \bigwedge_{i \in G} a_n(i,j)$, all agents in G are permitted to execute action a to agent j. When we write "*agent i is permitted to execute action a_n to agent j*", it means that agent i is allowing j to perform an action a_n which directly involves i, e.g. $P^{Alice}_{Bob} friendRequest$ would mean that Bob is allowed to send a friend request to Alice.

VI.2.2 Semantics of $\mathcal{KBL}_{\mathcal{T}}$

$\mathcal{KBL}_{\mathcal{T}}$ formulae will be interpreted over traces of social network models. These models are defined as social graphs [27] with agents, their knowledge bases and privacy policies, and a first-order relational structure.

Definition 2 (Social Network Model [67]). *Given a set of privacy policies* Π*, and a finite set of agents* $Ag \subseteq \mathcal{AU}$ *from a universe* $\mathcal{AU}$*; a* social network model (SNM) *is a social graph of the form* $\langle Ag, \mathcal{A}, KB, \pi \rangle$*, where*

- *Ag is a nonempty finite set of* nodes *representing the agents of the social network;*
- *$\mathcal{A}$ is a first-order relational structure over the SNM, consisting of a set of predicate symbols, function symbols and constant symbols interpreted over a domain from a set $\{D_o\}_{o \in \mathcal{D}}$, where $\mathcal{D}$ is a set of indexes for domains;*
- *$KB : Ag \rightarrow 2^{\mathcal{F}_{\mathcal{KBL}}}$ is a function returning the set of accumulated knowledge for each agent, stored in what we call the* knowledge base *of the agent. We write KB_i to denote $KB(i)$;*
- *$\pi : Ag \rightarrow 2^{\Pi}$ is a function returning the set of privacy policies of each agent. We write π_i for $\pi(i)$.*

The knowledge base KB_i of each agent i contains the explicit knowledge that the agent has. Agents not only posses this explicit knowledge, but also anything that can be derived using the **S5** axiomatisation of epistemic logic [29] from the explicitly formulae in their knowledge bases.

Definition 3. *A* derivation *of a formula $\varphi \in \mathcal{F}_{\mathcal{KBL}}$, is a finite sequence of formulae $\varphi_1, \varphi_2, \ldots, \varphi_n = \varphi$ where each φ_i, for $1 \leq i \leq n$, is either an instance of the axioms or the conclusion of one of the derivation rules of the* ***S5*** *axiomatisation which premises have already been derived, i.e., it appears as φ_j with $j < i$.*

Given a set of formulae $\Gamma \in 2^{\mathcal{F}_{\mathcal{KBL}}}$, we write $\Gamma \vdash \varphi$ to denote that φ can be derived from Γ.

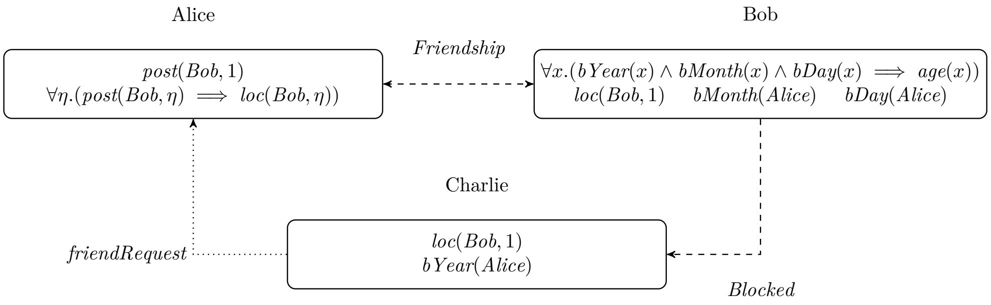

Alice
Bob
Friendship
$post(Bob, 1)$
$\forall\eta.(post(Bob,\eta) \implies loc(Bob,\eta))$
$\forall x.(bYear(x) \wedge bMonth(x) \wedge bDay(x) \implies age(x))$
$loc(Bob, 1)$ $bMonth(Alice)$ $bDay(Alice)$
Charlie
$loc(Bob, 1)$
$bYear(Alice)$
friendRequest
Blocked

Figure VI.1: Example of Social Network Model

Example 1. *Let SN be an SNM consisting of three agents Alice, Bob and Charlie, $Ag = \{Alice, Bob, Charlie\}$; the friend request action, $\Sigma = \{friendRequest\}$; and the connections Friendship and Blocked, $\mathcal{C} = \{Friendship, Blocked\}$.*

Fig. VI.1 shows a graphical representation of SN. In this model the dashed arrows represent connections. Note that the Friendship connection is bidirectional, i.e., Alice is friend with Bob and vice versa. On the other hand, it is also possible to represent unidirectional connections, as Blocked: in SN Bob has blocked Charlie. Permissions are represented using a dotted arrow. In this example, Charlie is able to send a friend request to Alice.

The predicates inside each node represent the agents' knowledge. Charlie and Bob have the predicate $loc(Bob, 1)$ inside the node, meaning that both know location number 1 of Bob. Besides predicates, agents' nodes may also contain $\mathcal{KBL}$ formulae that may increase the knowledge of the agents. For instance, Alice knows $loc(Bob, 1)$ implicitly. The rule Modus Ponens *can be applied to $\forall\eta.post(Bob,\eta) \implies loc(Bob,\eta)$ (included in Alice's knowledge base). This formula states that if Alice has access to a post of Bob, she can infer his location. Alice has access to $post(Bob, 1)$, therefore she can infer $loc(Bob, 1)$.* □

We use *traces* to capture the evolution of a social network. A trace is a sequence of pairs consisting of an SNM together with a *timestamp*. A timestamp t is a natural number representing the number of milliseconds elapsed since January 1, 1900, 00:00:00.000. We could have chosen a large number of equally good starting points; there is no specific reason for choosing 1900. We will use $\mathbb{T}$ to denote the set of timestamps. We will also use a more human-readable format of `YYYY-MM-DD hh:mm:ss.sss` for individual timestamps, optionally skipping the time part – then we assume it defaults to 00:00:00.000. Based on this, 2016-03-26 10:59:08.234 or 2000-01-01 (which represents

2000-01-01 00:00:00.000) or 1950-12-10 15:35:00.474 are all valid timestamps.

The intuitive meaning of a trace is that each SNM is a snapshot of the social network at point t, as if we froze the network, along with the knowledge and relationships between its agents, at that moment. In addition, we also demand the trace to be finite.

Definition 4 (Trace)**.** *A* trace σ *is a finite sequence* $\sigma = \langle (SN_0, t_0), (SN_1, t_1), \ldots, (SN_k, t_k)\rangle$ *such that, for all* $0 \leq i \leq k$*,* $SN_i \in \mathcal{SN}$ *and* $t_i \in \mathbb{T}$*.*

The (untimed) evolution of a social network was formalised in [67] using a transition relation $SN \xrightarrow{e} SN'$ where SN and SN' are SNMs and e is an event from a set EVT (the set of all the events that can be executed in the social network). Here we extend this transition relation with timestamps, formally $\rightarrow \subseteq \mathcal{SN} \times EVT \times \mathbb{T} \times \mathcal{SN}$. The details of how the elements of the SNM are changed because of the execution of the event are formally described in [67], but they are not relevant for the purpose of this paper. In order to explicitly define which event led to a given SNM in a trace, we sometimes write $SN_0 \xrightarrow{e,t_1} SN_1$ to denote $\langle (SN_0, t_0), (SN_1, t_1)\rangle$ where $SN_0, SN_1 \in \mathcal{SN}$, $e \in EVT$ and $t_0, t_1 \in \mathbb{T}$.

We also introduce the notion of *well-formed trace*, as being a trace satisfying the following two conditions. First, timestamps are strictly ordered from smallest to largest. Second, the transition from (SN_n, t_n) to (SN_{n+1}, t_{n+1}) occurs due to the execution of an event $e \in EVT$ at time t_{n+1}.

Definition 5 (Well-formed traces)**.** *Let* $\sigma = \langle (SN_0, t_0), (SN_1, t_1), \ldots, (SN_k, t_k)\rangle$ *be a trace.* σ *is* well-formed *if the following two conditions hold:*

i) Let $n \in \mathbb{N}$*. Then for any* i, j *such that* $0 \leq i, j \leq n$ *and* $i < j$*, it is the case that* $t_i < t_j$*.*

ii) Let $n \in \mathbb{N}$*. For all* $(SN_n, t_n), (SN_{n+1}, t_{n+1}) \in \sigma$*, it is the case that* $SN_n \xrightarrow{e,t_{n+1}} SN_{n+1}$ *where* $e \in EVT$*.*

We use $\mathcal{WFT}$ to denote the set of all well-formed traces. To make it easier to reason about the attributes of the whole trace, based on the attributes of individual networks, we use the following shortcuts to refer to the SNMs in a trace: $\sigma[t]$ denotes the SNM SN such that $(SN, t) \in \sigma$; given $t_1, t_2 \in \mathbb{T}$ such that $t_1 \leq t_2$, $\sigma[t_1..t_2]$ represents the well-formed subtrace of σ starting with the smallest $t \geq t_1$ such that $(SN, t) \in \sigma$, and ending with the largest $t \leq t_2$ such that $(SN, t) \in \sigma$.

Often we need to refer to the components of a specific SNM in a trace. For that purpose we use a superscript in the components of SNMs to indicate the SNM of the trace to which the element belongs. For example, given σ and t, $Ag^{\sigma[t]}$ stands for the

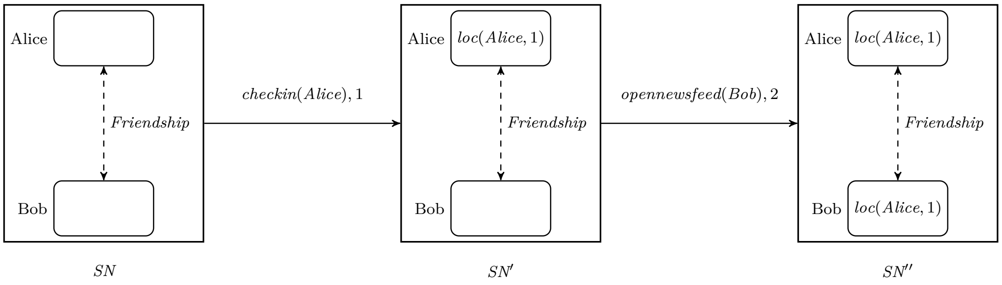


Figure VI.2: Example of a Trace of SNMs

set of agents in $\sigma[t]$, $D_o^{\sigma[t]}$ for a domain in $\mathcal{A}^{\sigma[t]}$ and $KB_i^{\sigma[t]}$ for i's knowledge base in $\sigma[t]$.

Furthermore, let Ag_σ contain all agents present in the trace, $Ag_\sigma = \bigcup_{(SN,t)\in\sigma} Ag^{SN}$, and $\mathbb{T}_\sigma$ denote the set of all timestamps associated with SNMs in the trace σ: $\mathbb{T}_\sigma = \{t \mid (SN,t) \in \sigma\}$.

Example 2. *Consider a social network with the event checkin, which discloses the location of users to all their friends, and the event opennewsfeed which retrieves all the posts, pictures, locations, etc., that a user has access to. Let* $\sigma = \langle (SN, 0), (SN', 1), (SN'', 2)\rangle$ *be the following well-formed trace* $SN \xrightarrow{checkin(Alice),1} SN' \xrightarrow{opennewsfeed(Bob),2} SN''$. *See Fig. VI.2 for a graphical representation of* σ.

It consists of the agents Alice and Bob, $Ag_\sigma = \{Alice, Bob\}$, *but new agents could be added by executing a sign up event. In this trace there are 3 timestamps,* $\mathbb{T}_\sigma = \{0, 1, 2\}$. *In the initial SNM, SN or* $\sigma[0]$, *Alice and Bob are friends,* $(Alice, Bob) \in C_{Friendship}^{\sigma[0]}$. *On the other hand, there is no knowledge neither in Alice's nor Bob's knowledge bases. At time 1 Alice discloses her location by performing a checkin event. Of course, she knows the location she just shared* $KB_{Alice}^{\sigma[1]} \vdash loc(Alice, 1)$ *(we use 1 as resource id to the location of Alice after she executes the checkin). Bob, however, did not check his newsfeed until time 2. Because of this it is at time 2 when he acquires the knowledge about Alice's location, i.e.,* $KB_{Bob}^{\sigma[1]} \nvdash loc(Alice, 1)$ *and* $KB_{Bob}^{\sigma[2]} \vdash loc(Alice, 1)$. $\square$

In what follows we define what means for a formula to be satisfied in a trace.

Definition 6 (Satisfaction)**.** *Given a well-formed trace* $\sigma \in \mathcal{WFT}$, *agents* $i, j \in Ag_\sigma$, *a finite set of agents* $G \subseteq Ag_\sigma$, *formulae* $\varphi, \psi \in \mathcal{F}_{\mathcal{TKBL}}$, $m \in \mathcal{C}$, $n \in \Sigma$, $o \in \mathcal{D}$, *and* $t, t' \in \mathbb{T}_\sigma$, *the* satisfaction relation $\vDash$ *is defined as shown in Table VI.1.*

We use a special agent called *environment* (or simply e) which defines the truth of predicates of the type $p(\vec{s})$. The environment's knowledge base (KB_e) contains

$\sigma, t \vDash \Box\varphi$	iff	for all $t' \in \mathbb{T}_\sigma$, $t' \geq t$, $\sigma, t' \vDash \varphi$
$\sigma, t \vDash \Diamond\varphi$	iff	there exists $t' \in \mathbb{T}_\sigma$, $t' \geq t$, such that $\sigma, t' \vDash \varphi$
$\sigma, t \vDash \neg\varphi$	iff	$\sigma, t \nvDash \varphi$
$\sigma, t \vDash \varphi \wedge \psi$	iff	$\sigma, t \vDash \varphi$ and $\sigma, t \vDash \psi$
$\sigma, t \vDash \forall x.\varphi$	iff	for all $v \in D_o^{\sigma[t]}$, $\sigma, t \vDash \varphi[v/x]$
$\sigma, t \vDash c_m(i,j)$	iff	$(i,j) \in C_m^{\sigma[t]}$
$\sigma, t \vDash a_n(i,j)$	iff	$(i,j) \in A_n^{\sigma[t]}$
$\sigma, t \vDash p(\vec{s})$	iff	$KB_e^{\sigma[t]} \vdash p(\vec{s})$
$\sigma, t \vDash K_i\varphi$	iff	$KB_i^{\sigma[t]} \vdash \varphi$
$\sigma, t \vDash L_i\varphi$	iff	$\sigma, t \vDash K_i\varphi$ and $\nexists t' < t$ such that $\sigma, t' \vDash K_i\varphi$
$\sigma, t \vDash D_G\varphi$	iff	$(\bigcup_{i \in G} KB_i^{\sigma[t]}) \vdash \varphi$

Table VI.1: Satisfaction Relation for $\mathcal{KBL}_\mathcal{T}$

all predicates which are true in the real world, e.g. *location*$(Alice) = $ "*Sweden*". In Table VI.1, $C_m \subseteq Ag \times Ag$ and $A_n \subseteq Ag \times Ag$ are binary relations used to interpret connection and actions predicates, respectively. We define $E_G^{k+1}\varphi$ as $E_G E_G^k\varphi$, where $E_G^0\varphi$ is equal to φ. We use $\varphi[v/x]$ to denote the usual capture-free substitution in first-order logic. For simplicity, we tacitly assume that each variable v is mapped to its own domain.

In their knowledge bases agents store all the information they get access to. At a given moment in time, the knowledge base of an agent contains all the information the agent have seen so far. Agents do not forget any piece of information since events updating knowledge can only add new formulae, but never remove them. This was formally defined in [67] where we describe the operational semantics rules which capture the evolution of SNMs. Formally, for any σ and $t \geq 1$ it always holds that

$$KB_i^{\sigma[t]} = KB_i^{\sigma[t-1]} \cup \Phi \tag{VI.1}$$

where $\Phi \in 2^{\mathcal{F}_{\mathcal{KBL}}}$ is a set of formulae representing the new knowledge that i learnt at time t. Given the above, it is easy to show that the latest knowledge base will always have the union of all the formulae that the agent had access to during the execution of the trace. Hence the following trivially follows from (VI.1):

$$\bigcup_{t' \leq t} KB_i^{\sigma[t']} = KB_i^{\sigma[t]}.$$

Therefore in order to check whether some piece of information is derivable from the knowledge base at some time t it suffices to check derivability from $KB_i^{\sigma[t]}$. On the other hand, learning is new knowledge (i.e., it was not known before) that is acquired at a given moment in time. As shown in Table VI.1, the difference with knowledge is that for learning we require that the new information cannot be derived in any knowledge base of the trace from a time previous to the one in which the formula is being evaluated. For instance, if Alice posts her location at time 3 and Bob is part of the audience of this post, then Bob has learnt Alice's location at time 3, formally $\sigma, 3 \vDash L_{Bob} loc(Alice)$. Moreover, it holds that Bob knows Alice's location for any timestamp greater than 3 or $\sigma, t \vDash K_{Bob} loc(Alice)$ for $t \geq 3$. Note that at time 3 Bob learnt Alice's location, but also knows it, $\sigma, 3 \vDash L_{Bob} loc(Alice) \wedge K_{Bob} loc(Alice)$.

It always holds that if the agents learn something, then they know it. We call this the *learning axiom*:

$$\text{L. } L_i\varphi \implies K_i\varphi.$$

Lemma 1. *The learning axiom is sound with respect to traces of SNMs.*

It is easy to show that knowing does not imply learning in general. For instance, if $KB_i^{\sigma[t']} \vdash \varphi$ for some $t' < t$ then we have that $\sigma, t \vDash K_i\varphi$ holds but $\sigma, t \vDash L_i\varphi$ does not hold.

Another assumption in our models is that of *perfect recall*. When agents in the system learn something, they know it forever. The *perfect recall axiom* is formally defined as

$$\text{PF. } K_i\varphi \implies \Box K_i\varphi.$$

Lemma 2. *The perfect recall axiom is sound with respect to traces of SNMs.*

Example 3. *Let σ be the trace introduced in Example 2. Now we can check whether Bob learns Alice's location after she performs the checkin action:*

$$\sigma, 0 \vDash \Box(friend(Alice, Bob) \wedge checkin(Alice) \implies \exists x.\Diamond L_{Bob} loc(Alice, x)) \qquad \text{(VI.2)}$$

The $\Box$ operator requires that the implication over it applies for all $t \in \mathbb{T}_\sigma$ such that $t \geq 0$, which in this example are 0, 1 and 2. The $\Diamond$ operator represents that whenever the premise holds, there will be a time in the future where Bob will learn a location of Alice. In order to determine whether it is satisfied over σ, we look at all the elements of the formula. The first conjunct of the premise, friend$(Alice, Bob)$, is already satisfied in SN and it remains true for all the trace, i.e., $(Alice, Bob) \in A^{\sigma[t]}_{Friendship}$ for $t \geq 0$. checkin$(Alice)$ is a predicate representing that Alice executed the event checkin. As mentioned, this general purpose predicates are check in the environment's knowledge base. It first becomes true in SN', i.e., checkin$(Alice) \in KB_e^{\sigma[t]}$ for $t \geq 1$. SN" is the resulting SNM after Bob opens his newsfeed. At this moment he learns Alice's location, *i.e., $KB^{\sigma[2]}_{Bob} \vdash loc(Alice, 1)$. Therefore, since the premise holds at time 1 and at time 2 Bob learns the location we conclude that (VI.2) holds.* □

Even though the formula of the previous example hold in σ, it does not guarantee that the location learnt by Bob is the one disclosed during Alice's checkin. This is because the existential quantification ranges over all the location identifiers. Nevertheless, (VI.2) stated that *a location* has been learnt, and indeed, it happened.

The model-checking problem for $\mathcal{KBL}$ formulae in SNMs is decidable [67]. The addition of the new modalities preserves this property for the new $\mathcal{KBL}_{\mathcal{T}}$.

Theorem 1 (Model-checking)**.** *Determining whether a formula $\varphi \in \mathcal{F}_{\mathcal{TKBL}}$ is satisfied in a trace $\sigma \in \mathcal{WFT}$ at a given timestamp $t \in \mathbb{T}_\sigma$, that is checking $\sigma, t \vDash \varphi$, is decidable.*

A proof of the theorem and analysis of its complexity is presented in Section VI.3.

VI.2.3 Timed $\mathcal{PPL}$

We would like to equip the users of a social network with additional power in defining their privacy policies. This is done by extending the original privacy policy language $\mathcal{PPL}$ with time fields, which enable the user to specify a possibly recurring time window frame in which their policy should be enforced. The basic form of the privacy policies is as follows $[\![\varphi \implies \neg\psi]\!]_{\text{a}}^{[\text{ start } | \text{ duration } | \text{ recurrence }]}$ or $[\![\neg\psi]\!]_{\text{a}}^{[\text{ start } | \text{ duration } | \text{ recurrence }]}$ if the policy has no conditions. The *start* field is mandatory, but *recurrence* and *duration* are optional. Table VI.2 shows the intervals $0, 1, \ldots, i$ where a policy must be enforced according to its parameters. If the *recurrence* field is not defined, then a policy should be only enforced in interval 0. We refer to the new language as *timed privacy policy language*, $\mathcal{PPL}_{\mathcal{T}}$.

In addition to the notion of a timestamp, we define a related notion of *duration.* A duration d is a positive natural number representing the number of milliseconds elapsed between two points in time. Simply put, it stands for the absolute difference of two timestamps $|t_2 - t_1|$, i.e., the time elapsed between t_1 and t_2. We will use a more human-readable format of durations. For instance, $d = 60000$ will be *1 minute*, $d = 2167236000$ will be *25 days, 2 hours and 36 seconds*, and so on.

The starting time of a policy is always a timestamp, which allows us to pinpoint a specific moment in (real) time from which the policy should be enforced. The other two fields are *duration* and *recurrence.* They are written in the duration format. These two fields define an offset from a specific time based on the first field. Table VI.2 shows how the three time fields work together to capture certain time windows. A privacy policy with the time fields [2016-16-02 14:00 | 6 hours] is meant to be enforced on February

Interval	Start	End
0	start	start + duration
1	start + recurrence	start + recurrence + duration
2	start + 2 recurrence	start + 2 recurrence + duration
3	start + 3 recurrence	start + 3 recurrence + duration
⋮	⋮	⋮
i	start + i recurrence	start + i recurrence + duration

Table VI.2: Time windows defined by the time fields of a policy

16, 2016, from 14:00 to 20:00. On the other hand, [2016-01-01 18:00 | 12 hours | 1 day] stands for every night (from 18:00 to 6:00), starting on January 1, 2016. We are now ready to define the general shape of formulae used in the privacy policy language.

Definition 7 (Syntax of $\mathcal{PPL}_{\mathcal{T}}$)**.** *Given agents $i, j \in Ag$, a nonempty set $G \subseteq Ag$, a timestamp s, duration d and recurrence r, a variable x, a formula $\varphi \in \mathcal{F}_{\mathcal{KBL}}$, the predicate symbols $c_m(i,j), a_n(i,j), p(\vec{s}) \in \mathcal{A}$ where $m \in \mathcal{C}$ and $n \in \Sigma$, the* syntax of the privacy policy language with time *$\mathcal{PPL}_{\mathcal{T}}$ is inductively defined as:*

$$
\begin{aligned}
\delta &::= \quad \delta \wedge \delta \mid \forall x.\delta \mid \tau \\
\tau &::= \quad [\![\neg\alpha]\!]_i^{[\ s\ ]} \mid [\![\neg\alpha]\!]_i^{[\ s\ /\ d\ ]} \mid [\![\neg\alpha]\!]_i^{[\ s\ /\ d\ /\ r\ ]} \mid \\
&\qquad [\![\varphi \implies \neg\alpha]\!]_i^{[\ s\ ]} \mid [\![\varphi \implies \neg\alpha]\!]_i^{[\ s\ /\ d\ ]} \mid \\
&\qquad [\![\varphi \implies \neg\alpha]\!]_i^{[\ s\ /\ d\ /\ r\ ]} \\
\alpha &::= \quad \alpha \wedge \alpha \mid \forall x.\alpha \mid c_m(i,j) \mid a_n(i,j) \mid \alpha\prime \\
\alpha\prime &::= \quad \gamma\prime \mid L_i\gamma \\
\gamma\prime &::= \quad K_i\gamma \mid D_G\gamma \\
\gamma &::= \quad \gamma \wedge \gamma \mid \neg\gamma \mid p(\vec{s}) \mid \gamma' \mid c_m(i,j) \mid a_n(i,j) \mid \forall x.\gamma
\end{aligned}
$$

Note that L_i cannot appear under the scope of a knowledge modality. We will denote the set of all formulae of $\mathcal{PPL}_{\mathcal{T}}$ as $\mathcal{F}_{\mathcal{TPPL}}$ and the set of all formulae created using the α category (the restrictions) as $\mathcal{F}^{\mathcal{R}}_{\mathcal{TPPL}}$. Now we introduce the notion of *conformance* of a privacy policy in a trace.

Definition 8 (Conformance)**.** *Given a well-formed trace $\sigma \in \mathcal{WFT}$, an agent $i \in Ag_\sigma$, formulae $\varphi \in \mathcal{F}_{\mathcal{TKBL}}, \alpha \in \mathcal{F}^{\mathcal{R}}_{\mathcal{TPPL}}$, $o \in \mathcal{D}$, and $\delta, \delta_1, \delta_2 \in \mathcal{F}_{\mathcal{TPPL}}$, the conformance*

$$\begin{array}{lll}
\sigma \vDash_C \delta_1 \wedge \delta_2 & \text{iff} & \sigma \vDash_C \delta_1 \wedge \sigma \vDash_C \delta_2 \\
\sigma \vDash_C \forall x.\delta & \text{iff} & \text{for all } v \in D_o^{\sigma[t]},\ \sigma \vDash_C \delta[v/x] \\
\sigma \vDash_C \llbracket \neg\alpha \rrbracket_{\mathrm{i}}^{[\ \mathrm{s}\ |\ \mathrm{d}\ |\ \mathrm{r}\ ]} & \text{iff} & \text{for all } c \in \mathbb{N}_0 \text{ such that} \\
 & & 0 \leq s + cr \leq \max(\mathbb{T}_\sigma), \\
 & & \sigma \vDash_C \llbracket \neg\alpha \rrbracket_{\mathrm{i}}^{[\ \mathrm{s\ +\ cr}\ |\ \mathrm{d}\ ]} \\
\sigma \vDash_C \llbracket \neg\alpha \rrbracket_{\mathrm{i}}^{[\ \mathrm{s}\ |\ \mathrm{d}\ ]} & \text{iff} & \sigma[s\ ..\ s+d], s \vDash \Box(\neg\alpha) \\
\sigma \vDash_C \llbracket \neg\alpha \rrbracket_{\mathrm{i}}^{[\ \mathrm{s}\ ]} & \text{iff} & \sigma[s\ ..\ ], s \vDash \Box(\neg\alpha) \\
\sigma \vDash_C \llbracket \varphi \Rightarrow \neg\alpha \rrbracket_{\mathrm{i}}^{[\ \mathrm{s}\ |\ \mathrm{d}\ |\ \mathrm{r}\ ]} & \text{iff} & \text{for all } c \in \mathbb{N}_0 \text{ such that} \\
 & & 0 \leq s + cr \leq \max(\mathbb{T}_\sigma), \\
 & & \sigma \vDash_C \llbracket \varphi \Rightarrow \neg\alpha \rrbracket_{\mathrm{i}}^{[\ \mathrm{s\ +\ cr}\ |\ \mathrm{d}\ ]} \\
\sigma \vDash_C \llbracket \varphi \Rightarrow \neg\alpha \rrbracket_{\mathrm{i}}^{[\ \mathrm{s}\ |\ \mathrm{d}\ ]} & \text{iff} & \sigma[s\ ..\ s+d], s \vDash \Box(\varphi \Rightarrow \neg\alpha) \\
\sigma \vDash_C \llbracket \varphi \Rightarrow \neg\alpha \rrbracket_{\mathrm{i}}^{[\ \mathrm{s}\ ]} & \text{iff} & \sigma[s\ ..\ ], s \vDash \Box(\varphi \Rightarrow \neg\alpha)
\end{array}$$

Table VI.3: Conformance Relation for $\mathcal{PPL}_\mathcal{T}$

relation $\vDash_C$ is defined as shown in Table VI.3.

We use $\mathbb{N}_0$ to denote the set of natural numbers including 0. The relation $\vDash_C$ is given in terms of the satisfaction relation $\vDash$. First, the recurrence parameter determines the time windows in which the policy must hold. This is done by getting the starting timestamps of each time window, i.e., from 0 to $\max(\mathbb{T}_\sigma)$. Given that we consider finite traces there will be a finite number of time windows. Afterwards, subtraces from the calculated starting timestamp plus the duration of the policy are checked. In particular, we use *box* to check that all SNMs satisfy the privacy policy. When the duration is not specified we check from the starting timestamp to the end of the trace.

Theorem 2 (Checking conformance)**.** *Determining whether a privacy policy $\varphi \in \mathcal{F}_{\mathcal{TPPL}}$ is in conformance with a trace $\sigma \in \mathcal{WFT}$, that is checking $\sigma \vDash_C \varphi$, is decidable.*

In what follows we show some examples of using $\mathcal{PPL}_\mathcal{T}$ to encode evolving privacy policies.

Example 4. *On Friday, April 15, 2016, Alice decides that she wants to keep her private life separate from her life as a graduate student. In a social network with privacy policies using $\mathcal{PPL}_\mathcal{T}$, she can keep her supervisor Bob from learning her location on weekends by defining the following privacy policy:*

$$\delta = \llbracket \neg L_{Bob} location(Alice) \rrbracket_{Alice}^{\mathit{[\ 2016\text{-}04\text{-}16\ |\ 2\ days\ |\ 1\ week\ ]}}$$

Given a trace of the social network Alice and Bob use, δ would be checked first in the subtrace from Saturday 16th, 00:00, to Monday 18th, 00:00, then again from Saturday 23rd, 00:00, to the end of Sunday 24th, and so on.

In order for the trace to be in conformance with δ, in each of these subtraces, the $\mathcal{KBL}_{\mathcal{T}}$ formula $\Box(\neg L_{Bob} location(Alice))$ needs to be satisfied. Based on the satisfaction relation (cf. Def. VI.1), this is only the case if the formula $\neg L_{Bob} location(Alice)$ is satisfied at every point of the subtrace. This, in turn, means that it must not be the case that $L_{Bob} location(Alice)$ is satisfied in any of the SNMs of the subtrace.

To determine whether $L_{Bob} location(Alice)$, we check that location(Alice) can be derived from Bob's knowledge base at time t which represents the timestamp of the SNM currently being checked. In other words, we have to determine whether Bob learnt (either directly, or by inference) location(Alice) at point t. As t is a time in one of the time windows δ is defined for (i.e., a weekend sometime after April 16th), Bob having access to this particular piece of knowledge would be a violation of Alice's policy, since it would mean Bob managed to learn Alice's location on a weekend. Note, however, that Bob learning Alice's weekend location at any point not *during a weekend is not considered a violation of the policy. This is due to the fact that the policy is not checked outside the time windows it is defined for.* □

Example 5. *Charlie will start a new one-month job on July 1st, 2016, and he would like to ensure that, during this period, when he is at home only his friends can learn about his posts. He achieves this by using the following policy of $\mathcal{PPL}_{\mathcal{T}}$*

$$\delta = \forall x.[\![home(Charlie) \wedge \neg friend(Charlie, x) \implies \neg L_x post(Charlie, text)]\!]_{Charlie}^{[\text{ 2016-07-01, 31 days }]}$$

The predicate home(Charlie) is checked by consulting the environment. Checking whether δ is violated ultimately boils down to checking whether all TSNMs in the trace, starting at 2016-07-01 00:00 and ending at 2016-08-01 00:00, satisfy the whole formula inside the policy. It should be noted, however, that the time when the post was actually posted is irrelevant; what matters is when the users learn about it. So if Charlie is working and her colleague Daniel somehow gains access to her post that was originally posted when she was at home, it is not a violation of either of the policies. □

Example 6. *A few months ago Facebook decided to provide a way for users to get over the end of a relationship in an easier way. It is carried out by limiting the information that shows up from the former partner [62]. For instance, pictures, videos or posts from the ex-partner do not appear in the newsfeed. This can be seen as a special set of privacy policies that apply when users breakup. Of course, they can be modelled using $\mathcal{PPL}_{\mathcal{T}}$.*

Consider the privacy policy "if we break up, then you can no longer learn about pictures I am tagged in". Let us say it is Frank who wants to enforce this and Eve is his current girlfriend. He is free to write the following in $\mathcal{PPL}_{\mathcal{T}}$:

$$\delta = \forall \eta . [\![brokenup(Frank, Eve) \wedge taggedin(Frank, \eta) \implies \neg L_{Eve} picture(\eta)]\!]^{[\,t\,]}_{Frank}$$

Here, too, checking whether δ is violated with respect to a trace means checking the contents of the policy in all SNMs in the trace, starting at time t. So if Eve gains access to a picture Frank is tagged in that is new to her (no matter when it was originally posted) when they are no longer together, Frank's policy will be violated. $\square$

VI.3 $\mathcal{KBL}_{\mathcal{T}}$ model-checking

In order to show the proof of Theorem 1, that is, the decidability of the model-checking problem for $\mathcal{KBL}_{\mathcal{T}}$, we present a naive model-checking algorithm which implements directly the semantics of $\mathcal{KBL}_{\mathcal{T}}$ in Table VI.1.

Lemma 3. *Let σ be a well-formed trace of length n, $t \in \mathbb{T}_\sigma$ be a timestamp, and $\varphi \in \mathcal{F}_{\mathcal{TKBL}}$ be a formula. Let q be maximum number of nested quantifiers in φ and d an upper-bound on the size of the domains for the quantifiers. There is an algorithm that determines, using $O(n \times |\varphi| \times d^q \times |Ag|)$ queries to the epistemic reasoning engine, whether $\sigma, t \vDash \varphi$.*

Proof. We first expand the universal quantifiers in φ by inductively transforming each subformula $\forall x.\varphi'$ into a conjunction with one conjunct $\varphi'[v/x]$ for each element v in the domain D. The resulting formula is quantifier free and has size $O(|\varphi| \times d^q)$ where d is a bound on the size of the domain and q is the maximum nested stack of quantifiers. Let $\varphi_1, \ldots, \varphi_m$ be the subformulas of the resulting formula, ordered respecting the subformula relation. An easy induction on $k < m$ shows that we can label—for every agent and at every step of the trace—, starting from the earliest time-stamp with either φ_k or $\neg\varphi_k$. We begin with the atomic part, ρ in Def. 1:

- Checking $c_m(e, f)$ and $a_n(e, f)$ can be performed in constant time, simply by checking the model at the given instant, for every agent.

- Checking $p(\overrightarrow{s})$ at a given instant t requires one query to the epistemic reasoning engine for $KB_e^{\sigma[t]}$ (for the environment agent e and time stamp t).

Then, for the epistemic part (ψ in Def. 1) we first resolve all operators except L_i:

- Checking $\psi_k = \neg\psi_j$ and $\psi_k = \psi_j \wedge \psi_i$ can be done in constant time for each instant t and agent i, using the induction hypothesis.
- Checking $K_i\psi_j$ requires one query to the epistemic engine for $KB_i^{\sigma[t]} \vdash \psi_j$ per instant.
- Checking $D_G\psi$ at a given instant can be done with $|Ag|$ queries for each instant.

The operator $L_i\psi$ is handled directly by checking the values of $K_i\psi$ in the present instant and t the previous instant t', which has been precomputed as whether the formula ψ is present or not in the label for agent i at instants t and t'.

Finally, for the temporal part (φ in Def. 1), we traverse the trace from the end to the beginning.

- Checking $\Box\varphi$ at instant t requires obtaining φ at t and $\Box\varphi$ at the next instant, using the temporal expansion $\Box\varphi \equiv \varphi \wedge \mathsf{X}\varphi$.
- Checking $\Diamond\varphi$ requires the same information, and hence can also be done in time linear in the lenght of the trace, using the temporal expansion $\Diamond\varphi \equiv \varphi \vee \mathsf{X}\varphi$.

It is easy to see that the semantics of $\mathcal{KBL}_\mathcal{T}$ is captured by the algorithm and the bounds of the theorem in the number of queries to the epistemic engine are met. $\Box$

Computing whether a formula φ can be derived from a collection of knowledge facts is a PSPACE-complete problem by Fagin *et al.* in [29], so reducing the number of queries is essential. Note that in the algorithm presented in Lemma 3 the number of tests to the epistemic engine to resolve a $L_i\varphi$ formula is reduced to one based on the fact that the knowledge of the agents grows monotonically and provided that (a) $K_i\varphi$ is memoized at every step, and (b) the sequence is visited in increasing order.

VI.3.1 Timestamping knowledge

An alternative solution to the algorithm in Lemma 3 is to add timestamps to formulae in the agents' knowledge bases. If Alice learns Bob's location at time 3, we say that $(loc(Bob), 3) \in KB_{Alice}$. It also allows us to differentiate between a formula that has been learnt twice. For example, Alice can also learn Bob's location at time 7, thus $(loc(Bob), 3), (loc(Bob), 7) \in KB_{Alice}$.

This approach allows us to incrementally remember when facts are learnt by the epistemic engine, but requires to know upfront the formula to model-check. This approach requires to update the notion of derivability (cf. Def. 6) to support timestamps. For simplicity, this time we define a *timed closure* function which computes all derivable formulae from an agent's knowledge base taking into account timestamps. We denote this closure function $Tcl\colon 2^{\mathcal{F}_{\mathcal{KBL}}\times\mathbb{T}} \rightarrow 2^{\mathcal{F}_{\mathcal{KBL}}\times\mathbb{T}}$, for the formal definition see Appendix Def. 9. Thus the semantics of K_i and L_i would be modified as follows

$$\begin{array}{lll} \sigma, t \vDash K_i\varphi & \text{iff} & (\varphi, t') \in \mathit{Tcl}(KB_i^{\sigma[t]}) \text{ where } t' \in \mathbb{T} \\ \sigma, t \vDash L_i\varphi & \text{iff} & (\varphi, t) \in \mathit{Tcl}(KB_i^{\sigma[t]}) \end{array}$$

The timestamps of the formulae that are added to the agents' knowledge base must grow monotonically. In the axioms and derivation rules of $\mathit{Tcl}(KB_i)$ (cf. Def. 9) when two different timestamps are involved in the derivation we always take the maximum for the derived formula. If one timestamp is involved we keep the value. It turns out to be enough to preserve the monotonicity of the derivations.

Lemma 4. *Time in $\mathit{Tcl}(KB_i)$ is monotonic.*

In order to make sure of the correctness of this modification we also require that $\mathit{Tcl}(KB_i)$ preserves the amount of knowledge that the agents can infer with respect to the untimed version $\vdash$.

Lemma 5. *For all $\varphi \in \mathcal{F}_{\mathcal{KBL}}$, $KB_i \vdash \varphi$ iff $(\varphi, t) \in \mathit{Tcl}(KB_i)$ where $t \in \mathbb{T}$.*

As expected adding timestamps to derivations keeps the problem decidable.

Lemma 6. *For all $\varphi \in \mathcal{F}_{\mathcal{KBL}}$ and $t \in \mathbb{T}$, determining whether $(\varphi, t) \in \mathit{Tcl}(KB_i)$ is decidable.*

VI.3.2 Further Optimizations

There are other optimisations that could be applied to obtain a more practical algorithm. The first observation is that the size d of the domains in a practical SNM can be very large, but the potentially interesing instantiations of a universal quantifier in a formula can typically be bound with a much smaller value. This is the case when the sub-formula within the quantifier is guarded by a limiting predicate, that is the antecedent A of the formula $\forall x.[A \Rightarrow B]$ is, for example, *friend*(*Charlie*, x). The only potential instantiations that make the formula true are those friends of *Charlie*. This set can be scoped much better than the whole population of the social network. Another future optmization is to leverage the proof tree of one derivation for a subsequent derivation. Consider the check at time t for $KB_i^{\sigma[t]} \vdash \varphi \wedge \psi$. In a later check $KB_i^{\sigma[t+1]} \vdash \psi$ we could leverage the proof at time t to instantaneusly determine that ψ is derivable. Of course, this approach requires to always keep the proof tree in memory, which might be problematic if the knowledge bases of the agents grow quickly.

VI.4 Related Work

To the best of our knowledge, this work is the first attempt to formalise privacy policies for social networks which depend on time. However, specifying and reasoning about temporal properties in multi-agent systems using epistemic logic have been subject of study for a long time. It began with the so called *interpreted systems* (IS). In [29] Fagin *et al.* introduce IS as a model to interpret epistemic formulae with temporal operators such as box and diamond. IS have been used for security analyses of multi-agent systems. For instance, IS have been used to formalise the notion of secrecy [38] and information-flow properties such as non-interference [5]. $\mathcal{KBL}_\mathcal{T}$ has similar semantics to IS, but with the difference that in IS perfect recall is not always assumed. It means that forgetful agents can be modelled, unlike in SNMs.

Recent research has been carried out in extending IS to be able to reason about past or future knowledge. In [8] Moses *et al.* extend K_i with a timestamp $K_{i,t}$. It makes possible to express properties such as "Alice knows at time 5 that Bob knew p at time 3", i.e., $K_{Alice,5}K_{Bob,3}\ p$. $\mathcal{KBL}_\mathcal{T}$ is not as expressive as that of Moses, since formulae in $\mathcal{KBL}_\mathcal{T}$ cannot talk about future of past knowledge. In $\mathcal{KBL}_\mathcal{T}$ we can only differentiate between learning and knowing. Nevertheless, in the knowledge bases proposed in Section VI.3 we include timestamps indicating when the agents learnt the information. Thus, we claim that formulae in Moses' language could be interpreted in those knowledge bases. Using an approach similar to Moses' epistemic logic would make it possible to define learning in terms of knowledge, since the language permits to syntactically specify the timestamp of the knowledge.

In [97] Woźna & Lomuscio present TCTLKD a combination of epistemic logic, CTL, a deontic modality and real time. Formulae in TCTLKD are more expressive than both of the languages we present in this paper $\mathcal{KBL}_\mathcal{T}$ and $\mathcal{PPL}_\mathcal{T}$. The models used to interpret formulae in TCTLKD are also more complex than the ones presented in this paper, since they are based on a semantics for a branching logic. They are a combination of timed automata and IS plus a an equivalence relation for modelling permission. For our purpose we do not need such a complex modelling power. Even though it has not been formally studied, we claim that the complexity of the model-checking problem of formulae in TCTLKD is much higher than that of $\mathcal{KBL}_\mathcal{T}$.

VI.5 Final Discussion

We have presented a novel privacy policy framework with support for dynamic recurrent privacy policies that depend on time. It has been done by extending $\mathcal{FPPF}$ [67, 71]. Concretely, we have extended the knowledge-based logic with the temporal modalities

$\square$, $\Diamond$, and the learning modality L_i resulting in $\mathcal{KBL}_\mathcal{T}$. A satisfaction relation to check formulae in $\mathcal{KBL}_\mathcal{T}$ has been defined as well. These elements correspond the intermediate tools to the enforcement of privacy policies that depend on time. Additionally, we have studied some properties regarding the relation between knowledge and learning. We have provided the language $\mathcal{PPL}_\mathcal{T}$ to express timed privacy policies and a conformance relation to check that the policies are not violated during the execution of a trace. Finally, we have proved that checking conformance of a privacy policy in $\mathcal{PPL}_\mathcal{T}$ and model-checking of $\mathcal{KBL}_\mathcal{T}$ formulae are decidable.

Yet there are some limitations in $\mathcal{PPF}_\mathcal{T}$. Firstly, in $\mathcal{KBL}_\mathcal{T}$ we cannot write formulae that describe knowledge at a given moment in time. This could be solved by introducing a knowledge modality which includes the timestamp of the knowledge we are interested in checking. It would make the learning modality derivable from knowledge. Secondly, we only check the privacy policies during the time window specified in the policy. This might not be as expressive as one might wish. Consider that Alice enables the following policy “Only my friends can know my pictures during the weekend” (P1). Let Bob be a friend of Alice. If Alice shares a picture on Saturday, Bob will have access to it. Imagine now that on Monday Alice unfriends Bob. At this moment P1 should be violated, because Alice is not a friend with Bob and Bob knows a picture of Alice during the weekend. To enforce this stronger privacy policy we need to extend the logic not only with timestamps in the modalities, but also in the predicates. It will also enable the possibility of having more precise models where the timestamps of predicates represent their own timestamp instead of the time when an agent learnt them.

There exists a prototype implementation of some of the policies of $\mathcal{FPPF}$ in the social network Diaspora* [22, 21]. We are currently investigating how to adapt our implementation to support $\mathcal{PPF}_\mathcal{T}$ privacy policies.

VI.A Appendix

VI.A.1 Formal definitions of timed knowledge bases

Definition 9 (Timed Closure of a Knowledge base). *Given the knowledge base of an agent i, KB_i, Tcl(KB_i) satisfies the following properties:*

a) For all $\varphi \in \mathcal{F}_{\mathcal{KBL}}$ and $t \in \mathbb{T}$, If $(\varphi, t) \in Tcl(KB_i)$ then $(\neg\varphi, t) \notin Tcl(KB_i)$,

b) Introduction and elimination rules for conjunction:

$\wedge_I$ - *If $(\varphi, t) \in Tcl(KB_i)$ and $(\psi, t') \in Tcl(KB_i)$, then $(\varphi \wedge \psi, \max(t, t')) \in Tcl(KB_i)$*

$\wedge_{E_1}$ - *If $(\varphi \wedge \psi, t) \in KB_i$, then $(\varphi, t) \in Tcl(KB_i)$,*

$\wedge_{E_2}$ - *Analogous to* $\wedge_{E_1}$ *but for* ψ,

c) If $(\varphi, t) \in Tcl(KB_i)$ *and* $(\varphi \implies \psi, t') \in Tcl(KB_i)$*, then* $(\psi, \max(t, t')) \in Tcl(KB_i)$,

d) If $(\varphi, t) \in Tcl(KB_i)$ *then* $(K_i\varphi, t) \in Tcl(KB_i)$,

e) If φ *is provable in the axiomatisation* ***S5*** *([29]) from* $Tcl(KB_i)$*, then* $\varphi \in Tcl(KB_i)$*. Formally:*

A1 - If φ *is an instance of a first-order tautology, then* $(\varphi, t^\top) \in Tcl(KB_i)$,

A2 - If $(K_i\varphi, t) \in Tcl(KB_i)$ *and* $(K_i(\varphi \implies \psi), t') \in Tcl(KB_i)$*, then* $(K_i\psi, \max(t, t')) \in Tcl(KB_i)$,

A3 - If $(K_i\varphi, t) \in Tcl(KB_i)$*, then* $(\varphi, t) \in Tcl(KB_i)$,

A4 - If $(K_i\varphi, t) \in Tcl(KB_i)$*, then* $(K_iK_i\varphi, t) \in Tcl(KB_i)$,

A5 - If $(\varphi, t) \notin Tcl(KB_i)$*, then* $(\neg K_i\varphi, t) \in Tcl(KB_i)$,

R1 - Modus ponens*, it is defined as c),*

R2 - If (φ, t) *is provable from no assumptions (i.e.,* φ *is a tautology) then* $(K_i\varphi, t) \in Tcl(KB_i)$,

C1 - $(E_G\varphi, t) \in Tcl(KB_i)$ *iff* $(\bigwedge_{i \in G} K_i\varphi, t) \in Tcl(KB_i)$,

C2 - $(C_G\varphi, t) \in Tcl(KB_i)$ *iff* $(E_G(\varphi \wedge C_G\varphi), t) \in Tcl(KB_i)$,

RC1 - If $(\varphi \implies E_G(\psi \wedge \varphi), t)$ *is provable from no assumptions, then* $(\varphi \implies C_G\psi, t) \in Tcl(KB_i)$,

D1 - $(D_{\{i\}}\varphi, t) \in Tcl(KB_i)$ *iff* $(K_i\varphi, t) \in Tcl(KB_i)$,

D2 - If $(D_G\varphi, t) \in Tcl(KB_i)$*, then* $(D_{G'}\varphi, t) \in Tcl(KB_i)$ *if* $G \subseteq G'$,

DA2-DA5 Properties A2, A3, A4 and A5, replacing the modality K_i *with the modality* D_G *for each axiom.*

Remark 1. *The rules* $\wedge_{E_1}$ *and* $\wedge_{E_2}$ *can only be applied if the formula* $\varphi \wedge \psi$ *was explicitly added to the knowledge base to avoid illegal updates of timestamps. If* $\wedge_I$ *was used to construct* $\varphi \wedge \psi$*, the following derivation would be possible. Given that* $(\varphi, 1) \in Cl(KB_i)$ *and* $(\psi, 4) \in Cl(KB_i)$*, by* $\wedge_I$ *we get* $(\varphi \wedge \psi, 4) \in Tcl(KB_i)$*. Now if we apply* $\wedge_{E_1}$ *we can derive* $(\varphi, 4) \in Tcl(KB_i)$*, which adds an updated copy of* φ *to the knowledge base of the agents. This update must be forbidden because it corresponds to unrealistic knowledge.*

Remark 2. *Tautologies in* $Tcl(KB_i)$ *have the special timestamp* $t^\top$*. It is used to represent any timestamp, i.e.,* $t = t^\top$ *for all* $t \in \mathbb{T}$*. When tautologies are used in a derivation involving other premises we will always take the timestamp of the premise that is not a tautology, it is formally guaranteed by defining* $\max(t, t^\top) = t$.

VI.A.2 Proofs

Lemma 1. *The axiom L is sound w.r.t. traces of SNMs.*

Proof. It trivially follows from $\vDash$ (cf. Def. 6). We show that for all $\sigma \in \mathcal{WFT}$, $t \in \mathbb{T}$ and $\varphi \in \mathcal{F}_{\mathcal{KBL}}$ the following holds $\sigma, t \vDash L_i\varphi \implies K_i\varphi$. Assume that $\sigma, t \vDash L_i\varphi$. By $\vDash$, it follows that $KB_i^{\sigma[t]} \vdash \varphi$. Therefore, by $\vDash$ we can conclude that $\sigma, t \vDash K_i\varphi$. $\square$

Lemma 2. *The perfect recall axiom is sound w.r.t. traces of SNMs.*

Proof. It trivially follows from $\vDash$ (cf. Def. 6). We show that for all $\sigma \in \mathcal{WFT}$, $t \in \mathbb{T}$ and $\varphi \in \mathcal{F}_{\mathcal{KBL}}$ the following holds $\sigma, t \vDash K_i\varphi \implies \Box K_i\varphi$. Assume that $\sigma, t \vDash K_i\varphi$. By $\vDash$, it follows that $KB_i^{\sigma[t']} \vdash \varphi$ such that $t' \in \mathbb{T}_\sigma$ and $t' \leq t$. By $\vDash$, $\sigma, t \vDash \Box K_i\varphi$ holds iff for all $t'' \in \mathbb{T}$ such that $t'' \geq t$. Since at t there exists a t' such that $t' \leq t$ and information cannot disappear from KB_i, for all $t'' \in \mathbb{T}_\sigma$ such that $t'' \geq t$, it holds $KB_i^{\sigma[t'']} \vdash \varphi$, therefore $\sigma, t \vDash \Box K_i\varphi$ holds. $\square$

Theorem 2. *Determining whether a privacy policy $\varphi \in \mathcal{F}_{\mathcal{TPPL}}$ is in conformance with a trace $\sigma \in \mathcal{WFT}$, that is checking $\sigma \vDash_C \varphi$, is decidable.*

Proof. It follows from the Def. 8. We show it by induction on the structure of the privacy policies. The base cases are $[\![\neg\alpha]\!]_i$ and $[\![\varphi \implies \neg\alpha]\!]_i$ with their corresponding time frames [s] and [s | d]. Since satisfaction of $\mathcal{KBL}_\mathcal{T}$ formulae is decidable (cf. Theorem 1), we need to establish decidability of computing the time frame. For [s] it is decidable since we consider finite traces and it reduces to checking satisfaction at time s of all the SNMs included in the finite subtrace from s to the end of the original trace. The case [s | d] is analogous to the previous one, but taking a subtrace of the original trace, which length is determined by the duration parameter. The inductive steps are [s | d | r], $\wedge$ and $\forall$. the case The case [s | d | r] splits the trace in time frames according to the recurrence parameter. Given that σ is finite, there is only a finite number of integers from 0 to $\max(\mathbb{T}_\sigma)$, hence the splitting is finite. Moreover, it reduces to the case [s | d], which by induction hypothesis is decidable. As in the proof of Theorem 1 the inductive steps for $\forall$ and $\wedge$ follow from their induction hypothesis. $\square$

Lemma 3. *Time in timed derivations is monotonic.*

Proof. It follows from the definition of $Tcl(KB_i)$ (cf. Def. 9). We show that none of the properties in Def. 9 generate a formula with a timestamp less than that of any of the premises applied in the derivation. Properties a), $\wedge_{E_1}$, $\wedge_{E_2}$, d), A1, A3, A4, A5, R2, C1, C2, RC1, D1, D2, and DA3-DA5 do not change the timestamp of the premise they are applied on. Therefore the timestamp of a formula derived using any of the previous properties will be equal to that of the premise(s). The remaining properties $\wedge_I$, c), A2, R1 and DA2 always take the maximum timestamp of the premises used to derive the conclusion, hence the timestamp of a derived formula will always be greater of equal to that of the premise(s) used in the derivation. Finally, we conclude that none of the properties in $Tcl(KB_i)$ produces a formula with a timestamp smaller to that of the premise(s) used in the derivation, hence time in these derivations is monotonic. □

Lemma 4. *For all $\varphi \in \mathcal{F}_{\mathcal{KBL}}$, $\varphi \in Cl(KB_i)$ iff $(\varphi, t) \in Tcl(KB_i)$ where $t \in \mathbb{T}$.*

Proof. It trivially follows from the definition of $Cl(KB_i)$ [66]. Given that Tcl (cf. Def. 9) is characterised by the exact same properties and since the operations regarding timestamps (i.e., take the maximum of two timestamps or keep the same timestamp) do not affect the derivations using the axioms and rules of Tcl, we conclude that both sets contain the same formulae. □

Lemma 5. *For all $\varphi \in \mathcal{F}_{\mathcal{KBL}}$ and $t \in \mathbb{T}$, determining whether $(\varphi, t) \in Tcl(KB_i)$ is decidable.*

Proof. Checking whether a formula $\varphi \in \mathcal{F}_{\mathcal{KBL}}$ is derivable under the axiomatisation **S5** is decidable [29]. By definition, the closure function Cl (and consequently Tcl) are **S5** maximal-consistent sets, hence checking whether $\varphi \in Cl(KB_i)$ is decidable. Therefore, the remaining question is whether the derivation algorithm for timestamps is decidable. For properties a), $\wedge_{E_1}$, $\wedge_{E_2}$, d), A3, A4, A5, R2, C1, C2, RC1, D1, D2, and DA3-DA5, it is trivially decidable since t is not updated. For $\wedge_I$, c), A2, R1 and DA2, it is also decidable since computing the maximum of two numbers is decidable. Finally, when φ is a tautology $(\varphi, \top) \in Tcl(KB_i)$, hence once it is established that φ is a tautology the value of t is irrelevant (since $t = \top$ for all $t \in \mathbb{T}$), and as mentioned, determining whether φ a tautology is decidable. Finally, we conclude that $(\varphi, t) \in Tcl(KB_i)$ is decidable. □

Chapter VII

Timed Epistemic Knowledge Bases for Social Networks

Raúl Pardo, César Sánchez and Gerardo Schneider

Abstract. We present an epistemic logic equipped with time-stamps in the atoms and epistemic operators, which allows to reason not only about information available to the different agents, but also about the moments at which events happens and new knowledge is acquired or deduced. Our logic includes both an epistemic and a belief operator, which allows to model the disclosure of information that may not be accurate.

Our main motivation is to model rich privacy policies in online social networks. *Online Social Networks* (OSNs) are increasingly used for social interactions in the modern digital era, which bring new challenges and concerns in terms of privacy. Most social networks today offer very limited mechanisms to express the desires of users in terms of how information that affects their privacy is shared. In particular, most current privacy policy formalisms allow only *static policies*, which are not rich enough to express timed properties like "*my location after work should not be disclosed to my boss*". The logic in this paper makes it possible to express rich properties and policies in terms of the knowledge available to the different users and the time of actions and deductions. Our framework can be instantiated for different OSNs, by specifying the effect of the actions in the evolution of the social network and in the knowledge disclosed to each agent.

We present an algorithm for deducing knowledge, which can also be instantiated with different variants of how the epistemic information is preserved through time. Our algorithm allows to model not only social networks with eternal information but also networks with ephemeral disclosures. Policies are modelled as formulae in the logic, which are interpreted over timed traces representing the evolution of the social network.

VII.1 Introduction

Online Social Networks also known as *Social Networking Sites*, like Facebook [28], Twitter [94] and Snapchat [86] have exploded in popularity in recent years. According to a recent survey [45] nearly 70% of the Internet users are active on social networks.

Some concerns, including privacy, have arisen alongside this staggering increase in usage. Several studies [56, 41, 51, 57] report that privacy breaches are growing in number. Currently, the most popular social networks do not offer mechanisms that users can use to guarantee their desired privacy effectively. Moreover, virtually all privacy policies are static and cannot express timing preferences, including referring to temporal points explicitly and policies that evolve in time.

In [69] we presented a privacy policy framework able to express dynamic privacy policies by introducing an explicit learning operator and time intervals in the semantics of policies. The framework consists of a *knowledge based logic* to characterise what users in the social network know, and a *privacy policy language*, based on the previous logic, where users can limit who can know their information and when. Policies and formulae in the logic are interpreted over *social network models* which faithfully represent the social graph of OSNs. The policy language allows for the representation of recurrent privacy policies, .e.g., *'During the weekend only my friends can see my pictures'*. Note that the previous policy only requires to activate the static policy *'only my friends can see my pictures'* during weekends.

Though quite expressive, a major restriction of the logic in [69] is that it does not have *time*, so one cannot express different instants at which an event happens and when knowledge is acquired. Moreover, the logic only includes a knowledge modality thus implicitly assuming that the information that users are told is true. This assumption is, however, not realistic in social networks as users may also have *beliefs* given that not all sources of information are truthful. Information that users disclose might be false. There exists a growing interest in the detection of fake news [91, 92, 93].

To address this issues we propose to define a logic that: i) is tailored for social networks, i.e., allows for expressing properties based on the social connections between users; ii) combines knowledge and belief modalities to differentiate between beliefs that might be false and true knowledge; iii) has time-stamps in modalities and atoms, thus enabling the possibility of referring to the time of information, e.g., a weekend location, and the moment when it was learnt.

There exist already some logics that include these elements—but separately. One line of work studies logics where the belief modality and atoms are time-stampted [100]. Unfortunately, this logic lacks of a knowledge modality, and is not tailored for social networks. Instead, it is defined to reason about AGM belief revision, which is a more general setting than privacy in social networks. A logic to reason about how beliefs spread out in Twitter has been introduced in [98]. This logic, though tailored for social networks, does not include time-stamps, so it cannot be used for reasoning about time. Finally, [39] propose an axiomatisation of epistemic logic which combines knowledge and belief, but again, it does not contain time-stamps in modalities or atoms.

In this paper we leverage the insights from previous work to define a logic that combines knowledge, belief and time and is tailored for the purpose of specifying *dynamic privacy policies* for social networks. We refer the reader to Section VII.5 for a more detailed comparison of related works.

More concretely, we extend [69] by enriching the logic with explicit time instants, both in the atoms and in the epistemic operators. In the resulting logic, one can refer to the instant at which some knowledge is inferred, for example about the knowledge of another agent at another instant. The expressive power of the new logic allows to derive the *learning* operator from the time-stamped knowledge.

Second, we equip the logic with *belief* operators, with the only restriction that agents cannot believe in something that they know is false. This allows the instantiation of the framework to OSNs where gossiping is allowed, that is, the spreading of potentially false information. Analogous to the learning operator, we derive the *accept* operator as the moment in which an agent start believing in something.

Third, we introduce the notion of *extended knowledge bases* which allow to answer queries of (temporal) epistemic formulas against the knowledge acquired during a sequence of events. The algorithm uses epistemic deductive reasoning starting from a set of axioms, which include the corresponding conventional axioms of epistemic reasoning populated for all instants. Depending on the desired instantiation of the framework, the axiom of perfect recall or weaker versions of it can be instantiated which allows to model knowledge acquisition in eternal OSNs like Facebook and ephemeral OSNs like Snapchat. When weaker versions are used, one can derive operators that capture when an agent stops knowing something (the *forget* operator) or stops believing in something (the *reject* operator).

Using extended knowledge bases we define the semantics of the logic used to construct dynamic privacy policies. This provides the means for a particular OSNs to enforce privacy policies, for example by blocking an event that would result in a violation of a user's policy. We illustrate with examples how our rich logic allows to express privacy policies and how agents can infer knowledge in different instantiations of the framework.

The rest of the paper is organised as follows. Section VII.2 presents the framework and Section VII.3 introduces the logic $\mathcal{KBL}_{\mathcal{RT}}$. Section VII.4 shows how to express privacy policies using $\mathcal{KBL}_{\mathcal{RT}}$. Section VII.4 shows how different existing OSNs can be modeled within our framework, including Snapchat. Finally, Section VII.5 presents related work and Section VII.6 includes concluding remarks.

VII.2 A Timed Privacy Policy Framework

We introduce a formal privacy policy framework for OSNs where properties regarding users knowledge and beliefs as well as time can be expressed. Our framework extends the framework in [69], in which temporal properties were expressed using temporal operators $\Box$ and $\Diamond$. Our solution allows to describing properties at concrete moments in time. We also introduce a modality for beliefs which allows to distinguish between information that might be inaccurate.

Our framework, called $\mathcal{PPF}_{\mathcal{RT}}$, consists of the following components:

1. A timed epistemic logic, $\mathcal{KBL}_{\mathcal{RT}}$, where modalities and predicates are timestamped.

2. Extended social graphs, called *social network models*, which describe the state of the OSN. These graphs contain the users or *agents* in the system and the relations between them, and their knowledge and beliefs. We use $\mathcal{SN}_{\mathcal{RT}}$ to denote the universe of social network models, and use the notion of trace of social network models to describe the evolution of the system.

3. A parameter $\omega \in \mathbb{N}$ which determines for how long users remember information, e.g., 5 seconds, 24 hours or indefinitely. Users can acquire new believes that challenge their current knowledge and believes, which requires a resolution. We illustrate this by a parameter β, with two possibilities *conservative* or *susceptible*, which specifies how users behave when they learn new beliefs which are inconsistent with their current beliefs.

4. A privacy policy language based on the previous logic.

VII.3 A Timed Knowledge Based Logic

$\mathcal{KBL}_{\mathcal{RT}}$ is a knowledge based first order logic which borrows modalities from epistemic logic [29], equips modalities and predicates with time-stamps, and allows quantifiers over time-stamps.

VII.3.1 Syntax

Let $\mathcal{T}$ be a *vocabulary* which consists of a set of predicate symbols, function symbols, and constant symbols. Predicate and functions symbols have some implicit arity. We assume an infinite supply of variables $x, y, \ldots$. Terms of the elements of $\mathcal{T}$ can be built as $s ::= c \mid x \mid f(\vec{s})$ where $\vec{s}$ is a tuple of terms respecting the arity of f.

Let $\mathbb{T}$ denote a set of *time-stamps*, which is required to be a non-Zeno totally ordered set, that is, there is a finite number of instants between any two given instants. We use time-stamps to mark pieces of information or to query the knowledge of the agents at specific points in time. We consider Ag be a set of agents, $\mathcal{D}$ a set of domains, and use EVT for set of events that can be executed in a social network. For instance, in Facebook, users can share posts, upload pictures, like comments, etc. The set of events that users can perform depends on the social network. Similarly, we use $\mathcal{C}$ and Σ to denote special sets of predicate symbols that denote connections (between agents) and permissions. We introduce the syntax of $\mathcal{KBL}_{\mathcal{RT}}$ as follows:

Definition 1 (Syntax of $\mathcal{KBL}_{\mathcal{RT}}$)**.** *Given agents $i, j \in Ag$ a time-stamp $t \in \mathbb{T}$, an event $e \in EVT$, a variable x, a domain $D \in \mathcal{D}$, predicate symbols $c_c^t(i,j), a_a^t(i,j), p^t(\vec{s})$ where $c \in \mathcal{C}$ and $a \in \Sigma$, the* syntax of the real-time knowledge-based logic *$\mathcal{KBL}_{\mathcal{RT}}$ is inductively defined as:*

$$\varphi ::= \rho \mid \varphi \wedge \varphi \mid \neg\varphi \mid \forall t \cdot \varphi \mid \forall x : D \cdot \varphi \mid K_i^t \varphi \mid B_i^t \varphi$$
$$\rho ::= c_c^t(i,j) \mid c_a^t(i,j) \mid p^t(\vec{s}) \mid occurred^t(e)$$

Given a nonempty set of agents $G \subseteq Ag$, the additional epistemic modalities are defined $S_G^t \varphi \triangleq \bigvee_{i \in G} K_i^t \varphi$, $E_G^t \varphi \triangleq \bigwedge_{i \in G} K_i^t \varphi$.

The epistemic modalities stand for: $K_i^t \varphi$, agent i knows φ at time t; $S_G^t \varphi$, someone in the group G knows φ at time t; $E_G^t \varphi$, everyone in the group G knows φ at time t. We use the following notation as syntactic sugar $P_i^j a^t \triangleq a(i,j,t)$, meaning that "*agent i is permitted to execute action a to agent j*". For example, $P_{Bob}^{Alice} friendRequest^5$ means that Bob is allowed to send a friend request to Alice at time 5. We will use $\mathcal{F}_{\mathcal{KBL}_{\mathcal{RT}}}$ to denote the set of all well-formed $\mathcal{KBL}_{\mathcal{RT}}$ formulae. The syntax introduces the following novel notions that have not been considered in other formal privacy policies languages such as [31, 12, 66, 69].

Time-stamped Predicates. Time-stamps are explicit in each predicate, including connections and actions. A time-stamp attached to a predicate captures those moment in time when that particular predicate holds. For instance, if Alice and Bob were friends in a certain time period, then the predicate $friend^t(Alice, Bob)$ is true for all t falling into the period, and false for all t outside. This can be seen as the *valid time* in temporal databases [87].

Separating Knowledge and Belief. Not all the information that users see in a social network is true. For instance, Alice may tell Bob that she will be working until late, whereas she will actually go with her colleagues to have some beers. In this example, Bob has the (false) belief that Alice is working.

Traditionally, in epistemic logic, the knowledge of agents consists on true facts. Potentially false information is regarded as beliefs [29]. The set of axioms **S5** characterise knowledge and the axioms of **KD45** characterise belief [29]. For $\mathcal{KBL}_{\mathcal{RT}}$ we combine both notions in one logic. In the following section we describe how to combine these two axiomatisations based on the results proposed by Halpern *et al.* in [39].

Time-stamped Epistemic Modalities. Time-stamps are also part of the epistemic modalities K and B. Using time-stamps we can refer to the knowledge and beliefs of the agents at different points in time. For example, the meaning of the formula $B_{Bob}^{20:00} loc^{19:00}(Alice, work)$ is that Bob beliefs at 20: 00 that Alice's location at 19: 00 is work.

Occurrence of Events. It is important to be able to determine when an event has occurred. Such an expressive power allows users to define policies that are activated whenever someone performs an undesired event. Examples of these policies are: "if Alice unfriends Bob, she is not allowed to send Bob a friend request" or "if a Alice denies an invitation to Bob's party, then she cannot see any of the pictures uploaded during the party."

Here we introduce $occurred^t(e)$ to be able to syntactically capture the moment when a specific event e occurred. A similar predicate was introduced by Moses *et al.* in [8] for analysing communication protocols.

VII.3.2 Semantics

Real-Time Social Network Models

We introduce formal models which allow us to reason about specific social network states at a given moment in time. These models leverage the information in the social graph [27]—the core data model in most social networks [30, 11, 61]. Social graphs include the users (or *agents*) and the relationships between them. Moreover, in our models we include a knowledge base for each agent, and the set of privacy policies that they have activated. We reuse the models defined for the previous version of this framework [69]. Nevertheless, the expressiveness of the privacy policies that can be enforced in $\mathcal{PPF}_{\mathcal{RT}}$ have substantially increased (see Section VII.4).

Definition 2 (Social Network Models). *Given a set of formulae $\mathcal{F} \subseteq \mathcal{F}_{\mathcal{KBL}_{\mathcal{RT}}}$, a set of privacy policies Π, and a finite set of agents $Ag \subseteq \mathcal{AU}$ from a universe $\mathcal{AU}$, a* social network model *(SNM) is a tuple $\langle Ag, \mathcal{A}, KB, \pi \rangle$, where*

- *Ag is a nonempty finite set of nodes representing the agents in the social network;*
- *$\mathcal{A}$ is a first-order structure over the SNM. As usual, it consists of a set of domains, and a set relations, functions and constants interpreted over their corresponding domain.*
- *$KB : Ag \to 2^{\mathcal{F}}$ is a function retrieving a set of knowledge of an agent—each piece with an associated time-stamp. The set corresponds to the facts stored in the knowledge base of the agent; we write KB_i. for $KB(i)$;*
- *$\pi : Ag \to 2^{\Pi}$ is a function returning the set of privacy policies of each agent; we write π_i for $\pi(i)$.*

In Def. 2, the shape of the relational structure $\mathcal{A}$ depends on the type of the social network under consideration. We represent the connections—edges of the social graph—and the permission actions between social network agents, as families of binary relations, respectively $\{C_i\}_{i \in \mathcal{C}} \subseteq Ag \times Ag$ and $\{A_i\}_{i \in \Sigma} \subseteq Ag \times Ag$ over the domain of agents. We use $\{D_i\}_{i \in \mathcal{D}}$ to denote the set of domains. The set of agents Ag is always included in the set of domains. We use $\mathcal{C}, \Sigma$ and $\mathcal{D}$ to denote sets of connections, permissions and domains, respectively.

Evolution of Social Network Models

The state of a social network changes by means of the execution of *events*. For instance, in Facebook, users can share posts, upload pictures, like comments, etc. The set of events that users can perform depends on the social network. We denote the set of events that can be executed in a social network as *EVT*. We use traces to capture the evolution of the social network. Each element of the trace is a tuple containing: a social network model, a set of events, and a time-stamp.

Definition 3 (Trace). *Given $k \in \mathbb{N}$, a trace σ is a finite sequence*

$$\sigma = \langle (SN_0, E_0, t_0), (SN_1, E_1, t_1), \ldots, (SN_k, E_k, t_k) \rangle$$

such that, for all $0 \leq i \leq k$, $SN_i \in \mathcal{SN}_{\mathcal{RT}}$, $E_i \subseteq EVT$, and $t_i \in \mathbb{T}$.

We define $\mathbb{T}_\sigma = \{t \mid (SN, E, t) \in \sigma\}$ to be the set of all the time-stamps of σ. We impose some conditions to traces so that they accurately model the evolution of social networks. We say that a trace is *well-formed* if it satisfies the following conditions:

Ordered time-stamps.. Time-stamps are strictly ordered from smallest to largest.

Accounting for Events.. The definition has to account for events being explicit in the trace. Let $\rightarrow$ be a transition relation defined as $\rightarrow \subseteq \mathcal{SN}_{\mathcal{RT}} \times 2^{EVT} \times \mathbb{T} \times \mathcal{SN}_{\mathcal{RT}}$. We have $\langle SN_1, E, t, SN_2 \rangle \in \rightarrow$ if SN_2 is the result of the set of events $E \in EVT$ happening in SN_1 at time t. Note that we allow E to be empty, in which case $SN_2 = SN_1$. We will use the more compact notation of $SN_1 \xrightarrow{E,t} SN_2$ where appropriate.

Events are independent.. For each $\xrightarrow{E,t}$ the set of events E must only contain independent events. Two events are independent if, when executed sequentially, the execution order does not change their behaviour. Consider the following two events: $post(Charlie, Bob, ``London")$ (Charlie shares a post containing Bob's location), and $friendRequest(Alice, Charlie)$ (Alice sends a friend request to Charlie). Independently of the order in which the previous events are executed the resulting SNM will have a new post by Bob, and Charlie will receive a friend request from Alice. On the other hand, consider now: $post(Charlie, Bob, ``London")$ and $disallowLoc(Bob)$ (Bob activates a privacy policy which forbids anyone to disclose his location). In this case, if $post(Charlie, Bob, ``London")$ is executed first, the resulting SNM will contain the post by Charlie including Bob's location. However, if $disallowLoc(Bob)$ occurs before $post(Charlie, Bob, ``London")$, Charlie's post would be blocked—since it violates Bob's privacy policy. These two events are not independent.

More formally,

Definition 4. *Given two events $e_1, e_2 \in EVT$, we say that e_1 and e_2 are independent iff for any two traces σ_1 and σ_2*

$$\sigma_1 = SN_1^0 \xrightarrow{\{e_1\},t} SN_1^1 \xrightarrow{\{e_2\},t'} SN_1^2$$

$$\sigma_2 = SN_2^0 \xrightarrow{\{e_2\},t} SN_2^1 \xrightarrow{\{e_1\},t'} SN_2^2$$

it holds $SN_1^0 = SN_2^0$ and $SN_1^2 = SN_2^2$.

We can now provide a formal definition of well-formed SNM traces.

Definition 5 (Well-Formed Trace). *Let*

$$\sigma = \langle (SN_0, E_0, t_0), (SN_1, E_1, t_1), \ldots, (SN_k, E_k, t_k) \rangle$$

be a trace. σ is well-formed *if the following conditions hold:*

1. *For any i, j such that $0 \leq i, j \leq k$ and $i < j$, it is the case that $t_i < t_j$.*
2. *For all i such that $0 \leq i \leq k-1$, it is the case that $SN_i \xrightarrow{E_{i+1},\, t_{i+1}} SN_{i+1}$.*

3. For all $e_1, e_2 \in E_i$ for $0 \leq i \leq k$, e_1 is independent from e_2.

We will use *TCS* to refer to the set of all well-formed $\mathcal{PPF}_{\mathcal{RT}}$ traces. In order to be able to syntactically refer to the previous or next social network model, given a concrete time-stamp, we assume that there exist the functions *predecessor* (*pred*) and *next* (*next*). *pred* : $\mathbb{T} \to \mathbb{T}$ takes a time-stamp and returns the previous time-stamp in the trace. Since the set of time-stamps is non-Zeno it is always possible to compute the previous time-stamp. Analogously, *next* : $\mathbb{T} \to \mathbb{T}$ takes a time-stamp and returns the next time-stamp in the trace. In $\langle (SN_0, E_0, t_0), (SN_1, E_1, t_1), (SN_2, E_2, t_2) \rangle$, $\mathit{pred}(t_1) = t_0$ and $\mathit{next}(t_1) = t_2$. We define predecessor of the initial time-stamp to be equal to itsel, i.e., $\mathit{pred}(t_0) = t_0$. Similarly, next of the last time-stamp of the trace is equal to itself, i.e., $\mathit{next}(t_2) = t_2$.

Modelling knowledge

It is not a coincidence that $\mathcal{KBL}_{\mathcal{RT}}$ formulae look very similar to those of the language $\mathcal{L}_n$ originally defined for epistemic logic [29]. We would like to provide users in our system with the same notion of knowledge. Traditionally, in epistemic logic, the way to model and give semantics to $K_i\varphi$ is by means of an undistinguishability relation which connects all world that an agent considers possible [29]. In particular, when talking about traces of events the framework used is *Interpreted Systems* (ISs). In ISs traces are called *runs*. A run describes the state of the system at any point in (discrete) time. An IS $\mathcal{I}$ is composed by a set of runs and a undistinguishability relation ($\sim_i$)—for each agent i—which models the states of the system that agents consider possible at any point in time. Determining whether an agent i knows a formula φ (written in the language of epistemic logic), for a run r at time m is defined as follows:

$$(\mathcal{I}, r, m) \vDash K_i\varphi \text{ iff } (\mathcal{I}, r', m') \vDash \varphi \text{ for all } (r, m) \sim_i (r', m')$$

where (r, m) and (r', m') represent states of $\mathcal{I}$.

Additionally, Fagin *et al.* proposed an alternative encoding to answer epistemic queries from a knowledge base consisting in a set of accumulated facts [29][Section 7.3]. Let *kb* be an agent representing a knowledge base that has been told the facts $\langle \psi_1, \ldots, \psi_k \rangle$ for $k \geq 1$ in run r at time t. It was shown in [29][Theorem 7.3.1] that the following are equivalent:

a) $(\mathcal{I}^{kb}, r, m) \vDash K_{kb}\varphi$.

b) $\mathcal{M}_n^{rst} \vDash K_{kb}(\psi_1 \wedge \ldots \wedge \psi_k) \implies K_{kb}\varphi$.

c) $(\psi_1 \wedge \ldots \wedge \psi_k) \implies \varphi$ is a tautology.

where $\mathcal{I}^{kb}$ is an IS which models the behaviour of *kb* and $\mathcal{M}_n^{rst} \vDash \varphi$ means that φ is valid in the Kripke models with an accessibility relation that is reflexive (r), symmetric (s) and transitive (t). The previous theorem holds not only for a system consisting in a single knowledge base, but systems including several knowledge bases.

This way of modelling knowledge is very suitable for our social network models. As mentioned earlier, in a social network model the users' knowledge is stored in their knowledge base. Therefore, by using the equivalence in [29][Theorem 7.3.1] we can determine whether a user knows a formula φ from the conjunction of all the formulae it has been told, formally, $\bigwedge_{\psi \in KB_i} \psi \implies \varphi$.

However, as mentioned earlier, $\mathcal{KBL}_{\mathcal{RT}}$ is not the same language as $\mathcal{L}_n$. Therefore we cannot directly apply [29][Theorem 7.3.1] to determine whether a user knows a fact φ. In the following we described an extended knowledge base which supports all the components of $\mathcal{KBL}_{\mathcal{RT}}$.

Extended Knowledge Bases

An *Extended Knowledge Base* (EKB) consists in a collection $\mathcal{KBL}_{\mathcal{RT}}$ formulae without quantifiers. All domains in a SNM are finite at a given point in time—the might grow as events occur. On the one hand, regular domains, i.e., D^t in $\mathcal{A}$ are assumed to be finite. Therefore, they can be easily unfolded as a finite conjunction. On the other hand, the time-stamps domain—though infinite in general—for a given trace $\mathbb{T}_\sigma$ will be finite because traces are finite. Hence EKBs can be populated with the explicit time-stamps values that moment in time. Later in this section, we introduce some axioms which will define how time-stamps are handled. In what follows we introduce the axioms EKBs use to handle knowledge and belief.

Derivations in EKBs. The information stored in an agent's EKBs along a trace determines her knowledge. At a concrete moment in time, an agent's EKB contains the explicit knowledge she just learnt. New knowledge can be derived from the explicit pieces of information in agents' EKBs. Derivations are not limited to formulae of a given point in time, but also can use old knowledge. A *time window*, or simply, window, determines how much old knowledge is included in a derivation. We write $\Gamma \vdash (\varphi, w)$ to denote that φ can be derived from Γ given a window w. We provide a set of deduction rules, *DR*, of the form

$$\frac{\Gamma \vdash (\varphi, w')}{\Gamma \vdash (\psi, w)}$$

meaning that, given the set of premises Γ, ψ can be derived with a window w from φ in a window w'.

Definition 6. *A* timed derivation *of a formula $\varphi \in \mathcal{F}_{\mathcal{KBL}}$ given a window $w \in \mathbb{N}$, is a finite sequence of pairs of formulae and windows, $\mathcal{F}_{\mathcal{KBL}} \times \mathbb{N}$, such that $(\varphi_1, w_1), (\varphi_2, w_2), \ldots, (\varphi_n, w_n) = (\varphi, w)$ where each φ_i, for $1 \leq i \leq n$, follows from previous steps by an application of a deduction rule of DR which premises have already been derived, i.e., it appears as φ_j with $j < i$, and $w_j \leq w_i$.*

In what follows we present the concrete derivation rules that can be used in EKBs to derive knowledge. We define deduction rules based on well studied axiomatisations of knowledge and belief together with rules to deal with knowledge propagation.

Knowledge and Belief in EKBs. In EKBs knowledge and belief coexist. So far, the definition of knowledge that we provided in the previous section only takes into account the axioms for knowledge. In particular, the axiomatisation **S5**. Thus, EKBs can use any of the **S5** axioms to derive new knowledge from the conjunction of explicit facts in the knowledge base.

Fagin *et al.* provided an axiomatisation for belief [29], the **KD45** axiomatisation. It includes the same set of axioms as **S5** —replacing K_i by B_i—except for the axiom $K_i\varphi \implies \varphi$ (A3). The difference between knowledge and belief is that believes do not need to be true—as required by A3 in knowledge. The requirement is that an agent must have *consistent* beliefs. It is encoded in the following axiom $\neg B_i\bot$ (axiom D).

We can summarise the last two paragraphs as follows: Whenever a formula of the form $K_i\varphi$ is encountered, axioms from **S5** can be applied to derive new knowledge, and if the formula is of the form $B_i\varphi$ axioms from **KD45** can be used instead. Nonetheless, we are missing an important issue: How do knowledge and belief relate to each other? To answer this question we use two axioms proposed by Halpern *et al.* in [39]: (L1) $K_i\varphi \implies B_i\varphi$ and (L2) $B_i\varphi \implies K_iB_i\varphi$.

L1 expresses that when users know a fact they also believe it. It is sound with respect to the definition of both modalities, since knowledge is required to be true by definition (recall axiom A3). This axiom provides a way to convert knowledge to belief. L2 encodes that when agents believe a fact φ they know that they believe φ. Thus adding an axiom which introduces knowledge from belief—more precisely, introduces knowledge about the beliefs.

However, the axioms from **S5**, **KD45** and L1, L2 need to be adapted to $\mathcal{KBL}_{\mathcal{RT}}$ syntax—which is the type of formulae supported by EKBs. In particular, the modalities need a time-stamp. All these axiomatisations are defined for models which represent the system in a concrete time. That is, given the current set of facts that users have, they can apply the axioms to derive new knowledge (at that time). To preserve this notion we will simply add the time-stamp t to all modalities. Intuitively, it models that

Knowledge axioms		**Belief axioms**		**Knowledge-Belief axioms**	
A1	All tautologies of first-order logic	K	$B_i^t\varphi \wedge B_i^t(\varphi \implies \psi) \implies B_i^t\psi$	L1	$K_i^t\varphi \implies B_i^t\varphi$
A2	$K_i^t\varphi \wedge K_i^t(\varphi \implies \psi) \implies K_i^t\psi$	D	$\neg B_i^t\bot$	L2	$B_i^t\varphi \implies K_i^tB_i^t\varphi$
A3	$K_i^t\varphi \implies \varphi$	B4	$B_i^t\varphi \implies B_i^tB_i^t\varphi$		
A4	$K_i^t\varphi \implies K_i^tK_i^t\varphi$	B5	$\neg B_i^t\varphi \implies B_i^t\neg B_i^t\varphi$		
A5	$\neg K_i^t\varphi \implies K_i^t\neg K_i^t\varphi$				

Table VII.1: EKB axioms for a trace σ for each $t \in \mathbb{T}_\sigma$.

Knowledge deduction rules axioms			
$\dfrac{\varphi \text{ is a first-order tautology}}{\Gamma \vdash (\varphi, w)}\ \text{(A1)}$	$\dfrac{\Gamma \vdash (K_i^t\varphi, w) \qquad \Gamma \vdash (K_i^t(\varphi \implies \psi), w)}{\Gamma \vdash (K_i^t\psi, w)}\ \text{(A2)}$	$\dfrac{\Gamma \vdash (K_i^t\varphi, w)}{\Gamma \vdash (\varphi, w)}\ \text{(A3)}$	
$\dfrac{\Gamma \vdash (K_i^t\varphi, w)}{\Gamma \vdash (K_i^tK_i^t\varphi, w)}\ \text{(A4)}$	$\dfrac{\Gamma \vdash (\neg K_i^t\varphi, w)}{\Gamma \vdash (K_i^t\neg K_i^t\varphi, w)}\ \text{(A5)}$		
Belief deduction rules			
$\dfrac{\begin{array}{c}\Gamma \vdash (B_i^t\varphi, w)\\ \Gamma \vdash (B_i^t(\varphi \implies \psi), w)\end{array}}{\Gamma \vdash (B_i^t\psi, w)}\ \text{(K)}$	$\dfrac{}{\Gamma \vdash (\neg B_i^t\bot, w)}\ \text{(D)}$	$\dfrac{\Gamma \vdash (B_i^t\varphi, w)}{\Gamma \vdash (B_i^tB_i^t\varphi, w)}\ \text{(B4)}$	$\dfrac{\Gamma \vdash (\neg B_i^t\varphi, w)}{\Gamma \vdash (B_i^t\neg B_i^t\varphi, w)}\ \text{(B5)}$
Premise deduction rule	**Knowledge-Belief deduction rules**		
$\dfrac{\varphi \in \Gamma}{\Gamma \vdash (\varphi, w)}\ (\textsc{Premise})$	$\dfrac{\Gamma \vdash (K_i^t\varphi, w)}{\Gamma \vdash (B_i^t\varphi, w)}\ \text{(L1)}$	$\dfrac{\Gamma \vdash (B_i^t\varphi, w)}{\Gamma \vdash (K_i^tB_i^t\varphi, w)}\ \text{(L2)}$	

Table VII.2: EKB deduction rules for a trace σ for each $t \in \mathbb{T}_\sigma$.

if users have some knowledge at time t they can derive knowledge using the previous axioms, and, this derived knowledge, belongs to the same time t. Table VII.1 shows the complete list of axioms that can be applied given a trace σ for each time-stamps $t \in \mathbb{T}_\sigma$.

In order for these axioms to be used in timed derivations we now express them as deduction rules as shown in Table VII.2. Since all derivations are performed for the same t they all share the same window w.

As mentioned earlier, an EKB models the knowledge and beliefs of a user at a given moment in time. We encode this notion by adding an explicit K_i^t to every formula in a user's EKB. Formally, we say that users in a trace σ are *self-aware* iff for all $t \in \mathbb{T}_\sigma$ If $\varphi \in EKB_i^{\sigma[t]}$ then $\varphi = K_i^t\varphi'$. By assuming this property we can syntactically determine the time t when some knowledge φ enters an EKB. In what follows we assume that all well-formed traces are composed by self-aware agents.

Example 1. *Consider the following EKB from a trace σ of an agent i at time t.*

$$\boxed{\begin{array}{c} K_i^t(\forall t' \cdot \forall j\colon Ag^{t'} \cdot event^{t'}(j, pub) \implies loc^{t'}(j, pub)) \\ K_i^t event^t(Alice, pub) \end{array}}$$

$$EKB_i^{\sigma[t]}$$

In this EKB i can derive using the axioms in Table VII.1 that Alice's location at time t is a pub, i.e., $loc^{t'}(Alice, pub)$. Here we show the steps to derive this piece of information. We recall that quantifiers are unfolded when added to the knowledge base. Therefore, given $\mathbb{T}_\sigma = \{t_0, t_1, \ldots, t\}$

$K_i^t \forall j\colon Ag^{t_0} \cdot event^{t_0}(j, pub) \implies loc^{t_0}(j, pub) \wedge$
$K_i^t \forall j\colon Ag^{t_1} \cdot event^{t_1}(j, pub) \implies loc^{t_1}(j, pub) \wedge$
$\ldots$
$K_i^t \forall j\colon Ag^{t} \cdot event^{t}(j, pub) \implies loc^{t}(j, pub)$

where each of these are also syntactic sugar, for instance, given $Ag^t = \{j_1, j_2, \ldots, j_n\}$ for $n \in \mathbb{N}$

$K_i^t event^t(j_1, pub) \implies loc^t(j_1, pub) \wedge$
$K_i^t event^t(j_2, pub) \implies loc^t(j_2, pub) \wedge$
$\ldots$
$K_i^t event^t(j_n, pub) \implies loc^t(j_n, pub)$

The predicate $event^t(j, pub)$ means that j attended an event at time t in a pub. The predicate $loc^t(j)$ means that j's location is a pub. Thus, the implication above encodes that if i's knows at t that if an agent is attending an event in a pub at time t, her location will be a pub. Moreover, i knows at time t that Alice is attending an event at the pub, $event^t(Alice, pub)$. As mentioned earlier, in epistemic logic, knowledge is required to be true. Therefore, $event^t(Alice, pub)$ must be a true predicate. From this we can infer that $Alice \in Ag^t$. Because of this, $K_i^t event^t(Alice, pub) \implies loc^t(Alice, pub)$ must also be present in $EKB_i^{\sigma[t]}$. Applying A2 to $K_i^t event^t(Alice, pub)$ and the previous implication we can derive $K_i^t loc^{t'}(j, pub)$ as required. $\square$

Handling time-stamps. In EKBs users can also reason about time. For instance, if Alice learns Bob's birthday she will remember this piece of information, possibly, forever. Nonetheless, this is not always true, some information is transient, i.e., it can change over time. Imagine that Alice shares a post including her location with Bob. Right after posting, Bob will know Alice's location—assuming she said the truth. However, after a few hours, Bob will not know for sure whether Alice remains in the same location. The most he can tell is that Alice was a few hours before in that location or that he believes that Alice's location is the one she shared. We denote the period of time in which some piece of information remains true as *duration.*

Different pieces of information might have different durations. For example, someone's birthday never changes, but locations constantly change. Duration also depends on the OSN. In Snapchat messages last 10 seconds, in Whatsapp status messages last 24 hours and in Facebook posts remain forever unless a user removes them. Due to these dependences we do not fix a concrete set of properties regarding time-stamps. Instead we keep it open so that they can be added when modelling concrete OSNs in $\mathcal{PPF}_{\mathcal{RT}}$. Concretely, the parameter ω introduced at the beginning of Section VII.2 corresponds to information duration for a particular OSN modelled in $\mathcal{PPF}_{\mathcal{RT}}$.

Using the window w—from timed derivations, see Def. 6—we define the following deduction rule encoding knowledge propagation. Given $t, t' \in \mathbb{T}_\sigma$ where $t < t'$:

$$\frac{\Gamma \vdash (K_i^t \varphi, w - (t' - t))}{\Gamma \vdash (K_i^{t'} \varphi, w)} \text{ (KR1)}$$

The intuition behind KR1 is that w is consumed every time knowledge is propagated. Imagine that Alice knows at time 1 the formula φ, $K^1_{Alice}\varphi$. Using KR1 in a derivation would allow us to derive, for instance, that she knows φ at a later time, e.g., at time 5, that is, $K^5_{Alice}\varphi$. Note that this derivation requires w to be at least 4, since Alice's knowledge of φ is propagated 4 units of time. As usual in when using this type rules, derivations can be described forwards or backwards. In the latter, the derivation starts with the conclusion of the rule—that is, we want to derive—and reduce the value of w every time we access old knowledge. In the former, we start from the premise of the rule and we increase w accordingly to derive the conclusion. The intuition and ways of deriving knowledge using KR1 are better illustrated with an example.

Example 2. *Consider the sequence of EKBs in Fig. VII.1 of an agent i from a trace σ where $\mathbb{T}_\sigma = \{0, \ldots, 4\}$. In this example we show the purpose of the window w when making derivations. Note that it is not possible to derive Alice's location only from the set of facts in a single knowledge base at a time t. Instead it is required to combine knowledge from different knowledge bases. We use the knowledge recall rule with different windows to access previous knowledge. As mentioned earlier, intuitively, w determines for how long agents remember information. Therefore, it is required to find an appropriate value for w that includes the sufficient knowledge—from all moments in time—to perform the derivation. In the figure, the red square marks the accessible knowledge for $w = 2$ and the blue square for $w = 3$.*

As can be seen in the trace, in order for i to derive $event^3(Alice, pub)$ she needs to *combine knowledge from* $EKB_i^{\sigma[0]}$ *and* $EKB_i^{\sigma[3]}$. *Let* $EKB_i^{\sigma} = \bigcup_{t\in\mathbb{T}_\sigma} EKB_i^{\sigma[t]}$. *First, we show how to construct a proof forwards, i.e., starting from the premises and a window of 0, move forward—by increasing the size of* w*—until the inference can be performed. In particular, we show that* $EKB_i^{\sigma} \vdash (K^3_{loc^3(Alice,pub)}, w)$ *for* $w \in \mathbb{N}$. *Let us start by applying the rule* PREMISE *with* $w = 0$*,*

$$EKB_i^{\sigma} \vdash (K_i^0 event^3(Alice, pub) \implies loc^3(Alice, pub), 0)$$

Recall that the quantifiers in the example are just syntactic sugar. They are replaced when added to the EKB. Now we use KR1 to combine this knowledge with knowledge at time 3*. In other words, we propagate knowledge from time 0 (*K_i^0*) to time 3 (*K_i^3*).*

$$\text{(KR1)}\ \frac{EKB_i^{\sigma} \vdash (K_i^0 event^3(Alice, pub) \implies loc^3(Alice, pub), 0)}{EKB_i^{\sigma} \vdash (K_i^3 event^3(Alice, pub) \implies loc^3(Alice, pub), 3)}$$

As stated in the rule the window has been incresed by 3, since $0 = 3 - (3 - 0)$*. We apply* PREMISE *again to obtain* $(EKB_i^{\sigma} \vdash K_i^3 event^3(Alice, pub), 3)$*. As in Example 1 by applying A2 in the previous statements we derive* $(EKB_i^{\sigma} \vdash K_i^3 loc^3(Alice, pub), 3)$*. This proof shows that i knows Alices location provided that agents remember information during 3 units of time.*

We show now that a window smaller than 3 makes this derivation impossible. Also we construct the proof backwards, i.e., starting from the conclusion we try to prove the required premises. Let us take a largest window smaller than 3, $w = 2$*. We try to show that* $EKB_i^{\sigma} \vdash (K_i^3 loc^3(Alice, pub), 2)$*. In order to prove we need to show the following:*

$$\text{(A2)}\ \frac{\begin{array}{c}(EKB_i^{\sigma} \vdash K_i^3 event^3(Alice, pub), 2)\\ EKB_i^{\sigma} \vdash (K_i^3 event^3(Alice, pub) \implies loc^3(Alice, pub), 2)\end{array}}{EKB_i^{\sigma} \vdash (K_i^3 loc^3(Alice, pub), 2)}$$

The first premise, $(EKB_i^{\sigma} \vdash K_i^3 event^3(Alice, pub), 2)$*, trivally follows by* PREMISE*. For the second premise we first try move one step back using KR1, i.e,*

$$EKB_i^{\sigma} \vdash (K_i^2 event^3(Alice, pub) \implies loc^3(Alice, pub), 1),$$

since there is no knowledge at time 2*, the previous statement cannot be proven. We apply again KR1 obtaining*

$$EKB_i^{\sigma} \vdash (K_i^1 event^3(Alice, pub) \implies loc^3(Alice, pub), 0).$$

Similarly, this statement cannot be proven. Note that the window is 0*. Intuitively, it means that we have access all knowledge that i remembers. Therefore, we have reach a*

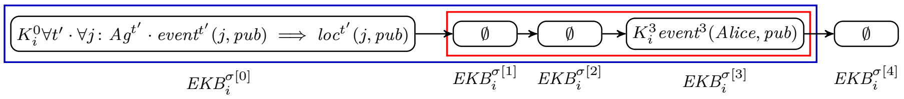


Figure VII.1: Sequence of EKBs of agent i for a trace σ where $\mathbb{T}_\sigma = \{0, \ldots, 4\}$

dead end in our proof tree. KR1 could be applied again, which would give

$$EKB_i^\sigma \vdash (K_i^1\, event^3(Alice, pub) \implies loc^3(Alice, pub), -1).$$

However, it would not be a valid timed derivation. In Def. 6 we require $w \in \mathbb{N}$. Finally, we conclude that $K_i^3 loc^3(Alice, pub)$ cannot be derived with a window of 2. $\square$

Belief propagation. Beliefs cannot be propagated as easily as knowledge. The reason is that, at some point in the future, new beliefs—contradicting the current set of beliefs of an agent—can enter an EKB. Therefore, we cannot simply increase the window during the derivation. Instead we propagate as long as beliefs are consistent.

As mentioned earlier, agents can be *conservative* or *susceptible.* It is specified in the parameter β of the framework (see Section VII.2). Conservative agents reject any new beliefs that contradict their current set of beliefs. On the contrary, susceptible agents always accept new beliefs and remove the old ones that contradict them. Here we present two different belief propagation algorithms which describe how agents behave when faced with a new belief which is contradictory.

Let σ be a trace with the following set of time-stamp, $\mathbb{T}_\sigma = \{t_0, \ldots, t_{n-1}, t_n\}$. We define $EKB_i^{\sigma[t_0,t_k]} = \bigcup_{t \in \{t_0,\ldots,t_k\}} EKB_i^{\sigma[t]}$ for $k \in \mathbb{N}$. Also, we introduce the event $enter(B_i^t \varphi)$ meaning that *belief φ enter i's knowledge base at time t.* We use this event to be able to identify the moment where a belief was inserted in an agent's EKB. As we will see in the following, determining when a belief enters an agent's knowledge base is crucial to propagating beliefs. Independently of the propagation rule of the agent, both algorithms will try to propagate the accumulated set of beliefs as long as they are inside the window w. Formally we define this set as $\Psi_c^{t_n}$ =

$$\{K_i^{t_n} B_i^{t_n} \psi \mid K_i^{t_{n-1}} B_i^{t_{n-1}} \psi \in EKB_i^{\sigma[t_{n-1}]} \text{ and } occurred^t(enter(B_i^t \varphi)) \in EKB_i^{\sigma[t_n - w, t_n]}\}$$

Intuitively, $\Psi_c^{t_n}$ includes all beliefs from the oldest knowledge base ($EKB_i^{\sigma[t-1]}$) that were inserted at a time t not older than allowed by the time window w. When a belief φ is inserted in an EKB, the fact $enter(B_i^t \varphi)$ is also included thus making the agent aware of the event of inserting a new belief.

Conservative agents. Conservative agents never change their beliefs. Therefore, when they start believing some fact, they will not accept its negation. Let be a trace $\sigma = SN_{t_{n-1}} \xrightarrow{\mathsf{e}(B_i^{t_n}\varphi),0} SN_{t_n}$ where at time t_n, agent i is told a belief $B_i^{t_n}\varphi$, represented by the event $\mathsf{e}(B_i^{t_n}\varphi)$. In what follows we describe how $EKB_i^{\sigma[t_n]}$ is updated, i.e., whether the belief is added or not. A conservative agent i updates her EKB as follows:

$$EKB_i^{\sigma[t_n]} = \begin{cases} \Psi_c^{t_n} \cup \{K_i^{t_n} B_i^{t_n}\varphi, \mathit{occurred}^{t_n}(\mathit{enter}(B_i^{t_n}\varphi))\} \text{ if } (EKB_i^{\sigma[t_0,t_{n-1}]} \cup \Psi_c^{t_n} \not\vdash B_i^{t_n}\neg\varphi, w) \\ \Psi_c^{t_n} \text{ otherwise} \end{cases}$$

All beliefs in $\Psi_c^{t_n}$ are propagated independently of the new belief ($B_i^{t_n}\varphi$) that the agent has been told—conservative agents never reject old beliefs. The new belief only enters i's EKB if it is consistent with her previous beliefs ($EKB_i^{\sigma[t_0,t_{n-1}]}$) and the ones to propagate ($\Psi_c^{t_n}$).

Susceptible agents. Susceptible agents always accept new beliefs. If new beliefs are contradictory to what they used to believe, they, simply, reject their old beliefs. Or, in other words, these beliefs are not propagated. However, sometimes agents need to choose which old beliefs they reject. For example, consider that $B_i^1\varphi$ and $B_i^1(\neg\varphi \vee \neg\psi)$. At time 2, a belief $B_i^2\psi$ enters i's EKB. Since i is susceptible, she accepts the new belief. Now she needs to choose whether to propagate $B_i^2\varphi$ or $B_i^2(\neg\varphi \vee \neg\psi)$—accepting both creates inconsistent beliefs. To address this issue we assume that there exists a total order among all beliefs to propagate $\Psi_c^{t_n} = \{\beta_0, \ldots, \beta_m\}$ for $m \in \mathbb{N}$ where $\beta_j < \beta_k$ if $j < k$. Intuitively, this order represents the order in which beliefs were told to the agent. Having $\beta_j < \beta_k$ means that the agent was told first β_j and later β_k. Consider again the trace $\sigma = SN_{t_{n-1}} \xrightarrow{\mathsf{e}(B_i^{t_n}\varphi),0} SN_{t_n}$ where at time t_n belief $B_i^{t_n}\varphi$ is about to enter $EKB_i^{\sigma[t_n]}$. We define the set of propagated beliefs for susceptible agents $\Gamma_c^{t_n} \subseteq \Psi_c^{t_n}$ as the largest set of beliefs such that the following conditions hold:

1. $EKB_i^{\sigma[t_0,t_{n-1}]} \cup \Gamma_s^{t_n} \not\vdash B_i^{t_n}\neg\varphi$;
2. If $\beta_j \notin \Gamma_c^{t_n}$ then $\left(EKB_i^{\sigma[t_0,t_{n-1}]} \cup \{\beta_k \in \Gamma_c^{t_n} | k > j\} \cup \{\beta_j\}\right) \vdash B_i^{t_n}\neg\varphi$.

Given the above, a susceptible agent updates her EKB as follows,

$$EKB_i^{\sigma[t_n]} = \Gamma_c^{t_n} \cup \{K_i^{t_n} B_i^{t_n}\varphi, \mathit{occurred}^{t_n}(\mathit{enter}(B_i^{t_n}\varphi))\}$$

Consistency is guaranteed by definition (see item (1) in the conditions for Ψ_s), since the rule states that any belief that contradicts φ will not be propagated.

Example 3. *At* 20: 00 *Bob receives a Facebook message from Alice telling him that she* *is at work. That is,*

$$\begin{aligned}EKB^{\sigma[20:00]}_{Bob} = \{&K^{20:00}_{Bob}B^{20:00}_{Bob}loc^{20:00}(Alice, work)\} \cup \\ &\{occurred^{20:00}(enter(B^{20:00}_{Bob}loc^{20:00}(Alice, work)))\}.\end{aligned}$$

At 22: 00 *Bob checks his Facebook timeline, and he sees a post of Charlie—who is a coworker of Alice—from* 20: 00 *saying that he is with all his coworkers—including Alice—in a pub having a beer. Assuming that at* 22: 00 *Bob still remembers his belief from* 20: 00*—i.e., the time window is larger than* 2 *hours for Facebook—this new information creates a conflict with Bob's beliefs. Note that information from Charlie's post is also concerned as belief since there is no way for Bob to validate it. Depending on the type of agent that Bob is there will be two possible updates in Bob's EKB.*

If Bob is a conservative agent, then

$$EKB^{\sigma[22:00]}_{Bob} = \{K^{22:00}_{Bob}B^{22:00}_{Bob}loc^{20:00}(Alice, work)\}$$

meaning that the new belief is rejected. Bob will continue believing that Alice is at work.

On the other hand, if Bob is a susceptible agent, he will add this new believe to his EKB and reject his old belief about Alice being at work, i.e.,

$$\begin{aligned}EKB^{\sigma[22:00]}_{Bob} = \{&K^{22:00}_{Bob}B^{22:00}_{Bob}loc^{20:00}(Alice, work)\} \cup \\ &\{occurred^{22:00}(enter(B^{22:00}_{Bob}loc^{20:00}(Alice, work)))\}.\end{aligned}$$

For all t such that $20:00 \leq t < 22:00$ *Bob believes that Alice's location at* 20: 00 *is work—due to belief propagation, and, after* 22: 00*, this belief does not propagate to avoid contradictions.* □

Semantics of $\mathcal{KBL}_{\mathcal{RT}}$ (RTKBL)

The semantics of $\mathcal{KBL}_{\mathcal{RT}}$ formulae is given by the following satisfaction relation $\vDash$.

Definition 7 (Satisfaction Relation)**.** *Given a well-formed trace $\sigma \in TCS$, agents $i, j \in Ag$, a finite set of agents $G \subseteq Ag$, formulae $\varphi, \psi \in \mathcal{F}_{\mathcal{KBL}_{\mathcal{RT}}}$, $m \in \mathcal{C}$, $n \in \Sigma$, $o \in \mathcal{D}$, a variable x, an event $e \in EVT$, and a time-stamp t, the* satisfaction relation *$\vDash\ \subseteq TCS \times \mathcal{F}_{\mathcal{KBL}_{\mathcal{RT}}}$ is defined as shown in Fig. VII.2.*

Predicates of type $occurred^t(e)$ are true if the event e is part of the events that occurred at time t in the trace. $\forall t$ quantifies over all the time-stamps in the trace $\mathbb{T}_\sigma$, which, as mentioned earlier, is a finite set. For the remaining domains, $\forall x : D^t$, the substitution is carried out over the elements of the domain at a concrete time t.

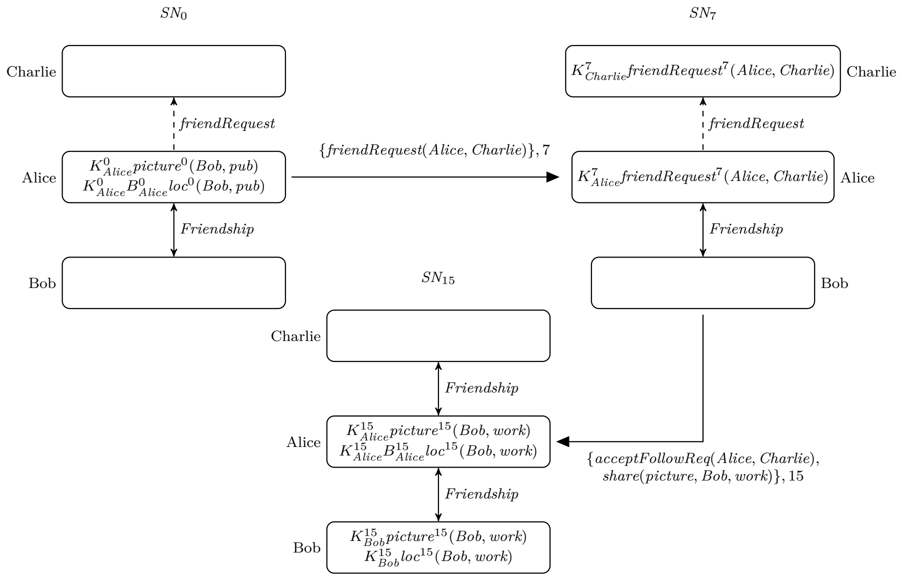
$$\begin{array}{lll}
\sigma \vDash \mathit{occurred}^t(e) & \text{iff} & (SN, E, t) \in \sigma \text{ such that } e \in E \\
\sigma \vDash \neg\varphi & \text{iff} & \sigma \nvDash \varphi \\
\sigma \vDash \varphi \wedge \psi & \text{iff} & \sigma \vDash \varphi \text{ and } \sigma \vDash \psi \\
\sigma \vDash \forall t \cdot \varphi & \text{iff} & \text{for all } v \in \mathbb{T}_\sigma,\ \sigma \vDash \varphi[v/t] \\
\sigma \vDash \forall x : D^t \cdot \varphi & \text{iff} & \text{for all } v \in D_o^{\sigma[t]},\ \sigma \vDash \varphi[v/x] \\
\sigma \vDash c_m^t(i,j) & \text{iff} & (i,j) \in C_m^{\sigma[t]} \\
\sigma \vDash a_n^t(i,j) & \text{iff} & (i,j) \in A_n^{\sigma[t]} \\
\sigma \vDash p^t\vec{s} & \text{iff} & p^t\vec{s} \in KB_e^{\sigma[t]} \\
\sigma \vDash K_i^t\varphi & \text{iff} & \bigcup_{\{t' \mid t' < t, t' \in \mathbb{T}_\sigma\}} KB_i^{\sigma[t']} \vdash (\varphi, \omega) \\
\sigma \vDash B_i^t\varphi & \text{iff} & \bigcup_{\{t' \mid t' < t, t' \in \mathbb{T}_\sigma\}} KB_i^{\sigma[t']} \vdash (B_i^t\varphi, \omega)
\end{array}$$

Figure VII.2: Satisfaction relation for $\mathcal{KBL}_{\mathcal{RT}}$

Remember that each individual domian D^t always contains a finite set of elements. However, the same domain at different points in time, e.g., D^t and $D^{t'}$, for any $t \neq t'$ might contain different number of elements. When checking connections $c_m^t(i,j)$ and actions $a_n^t(i,j)$ at time t, we check whether the corresponding relation— $C_m^{\sigma[t]}$ and $A_n^{\sigma[t]}$, correspondingly—of the SNM at time t contains the pair of users in question. Checking whether a predicate of type $p^t\vec{s}$ holds is equivalent to looking into the knowledge base of the environment at time t. The environment's knowledge base contains all predicates that are true in the real world at a given moment in time. For example, "it is raining in Gothenburg at 19:00" $\mathit{rain}^{19:00}(\mathit{Gothenburg})$ or "Alice's location at 20:00 is Madrid" $\mathit{loc}^{20:00}(\mathit{Alice}, \mathit{Madrid})$. Determining whether an agent i knows φ at time t, $K_i^t\varphi$, translates to checking whether φ can be derived from i's knowledge base at time t. In order to determine whether a user believes φ at time t, $B_i^t\varphi$, we check whether $B_i^t\varphi$ is derivable from the knowledge base of the user at time t given the parameter ω of the framework. This way of defining belief is based on the fact that agents are aware of their beliefs, recall axiom (L2) in Table VII.1. Therefore if a user believes φ, i.e., $B_i^t\varphi$, then she must also know that she believes it $K_i^tB_i^t\varphi$. Intuitively, $KB_i^{\sigma[t]} \vdash \varphi$ means that "user i knows φ at time t", given its equivalence to the knowledge operator. As expected, knowledge derivations are also limited by the parameter ω.

The intuition behind the previous definitions is better illustrated in the following example.

Example 4 (Snapchat)**.** *In this example we model the OSN Snapchat. In Snapchat there are two main events that users can perform: i) Connect through a friend relation; ii) share timed messages (which last up to 10 seconds and can include text and/or a picture) with their friends. Fig. VII.3 shows an example trace for Snapchat. The trace*



SN_0
Charlie
$friendRequest$
Alice
$K^0_{Alice}picture^0(Bob, pub)$
$K^0_{Alice}B^0_{Alice}loc^0(Bob, pub)$
$Friendship$
Bob
$\{friendRequest(Alice, Charlie)\}, 7$
SN_7
$K^7_{Charlie}friendRequest^7(Alice, Charlie)$
Charlie
$friendRequest$
$K^7_{Alice}friendRequest^7(Alice, Charlie)$
Alice
$Friendship$
Bob
SN_{15}
Charlie
$Friendship$
Alice
$K^{15}_{Alice}picture^{15}(Bob, work)$
$K^{15}_{Alice}B^{15}_{Alice}loc^{15}(Bob, work)$
$\{acceptFollowReq(Alice, Charlie),$
$share(picture, Bob, work)\}, 15$
$Friendship$
Bob
$K^{15}_{Bob}picture^{15}(Bob, work)$
$K^{15}_{Bob}loc^{15}(Bob, work)$

Figure VII.3: Example of a Snapchat trace

consists of: A common set of agents $Ag = \{Alice, Bob, Charlie\}$. Since Ag does not change we avoid using the superindex indicating the time-stamp of the domain. Three SNMs SN_0, SN_7 and SN_{15}. The subindex of the SNMs indicates their time-stamp.

At time 0, Alice and Bob are friends, i.e., $friendship^0(Alice, Bob)$. This is represented by including the pair $(Alice, Bob)$ in the relation $Friendship^{\sigma[0]}$, drawn in the picture as an arrow between Alice and Bob in SN_0. This relation between Alice and Bob does not change in σ. Moreover, in SN_0, Alice can send a friend request to Charlie. It is depicted as an outgoing dashed arrow from Alice's node to Charlie's. Thus, $\sigma \models P^{Charlie}_{Alice}friendRequest^0$ holds. Moreover, Alice knows that there is a picture of Bob at the pub, $picture^0(Bob, pub)$. On the other hand, she believes *that his location is the pub, $loc^0(Bob, pub)$. The reason for this is because she cannot verify that the picture has not been modified or she cannot precisely identify the location. However, the existence of $picture^0(Bob, pub)$ can be verified since it is a picture that Alice can see in the OSN.*

At time 7, Alice sends a friend request to Charlie. Though in this paper we do not discuss modelling the behaviour of events, we assume for this example that Alice can perform this events since he is explicitly permitted.[1] *After the execution of the*

[1]We refer the reader to [67] for the definition of operational semantics rules for $\mathcal{PPF}$ which describe the behaviour of the events in the OSN.

event both agents know friendRequest^7*(Alice, Charlie). Note that this event produces knowledge. This is because the agents can verify that the friend request has occurred.*

Lastly, at time 15*, Charlie accepts Alice's request and Bob shares a picture at work. Note that these two events are independent. If they were to be executed sequentially, independently of their order, the final SNM would always be equal to* SN_{15}*. After Bob's accepting Alice's request* $(Alice, Charlie) \notin FriendRequest^{\sigma[15]}$*, and* $(Alice, Charlie) \in Friendship^{\sigma[15]}$*. That is, Alice cannot send more friend requests to Charlie, and now they have become friends. Furthermore, both, Alice and Bob know that Bob shared a picture at work. Note that, in this case, Bob also knows that his location is work. Nevertheless, Alice believes it.*[2] *The reason is that, unlikely Bob, Alice cannot confirm that Bob's location is work.*

We mentioned that Snapchat messages last for up to 10 seconds. Let us assume w.l.o.g. that all messages last 10 seconds, i.e., $\omega = 10$*. Given that, in* σ*, Alice remembers Bob picture from 0 to 10. That is,*

$$\sigma \vDash \forall t \cdot 0 \leq t \leq 10 \implies K^t_{Alice} picture^0(Bob, pub)$$

Similarly, her belief about Bob location, $picture^0(Bob, pub)$*, vanishes at time 10. Note also that, when Charlie accepts Alice's friend request, he still knows (or remembers) that Alice sent it. In Snapchat friend request are permanent, but in* $\mathcal{PPF}_{\mathcal{RT}}$ *we can choose whether friend request disappear after a few seconds. It can be done by requiring that the agent knows that a friend request occurred in order to accept it. In such a case, in* σ*, after time* 18 *Charlie would not be able to accept Alice's request.*

As mentioned earlier, the purpose of $\mathcal{PPF}_{\mathcal{RT}}$ *main goal is expressing privacy policies. They can be expressed as a formulae in* $\mathcal{KBL}_{\mathcal{RT}}$*. Let Ag and Locs be domains of agents and locations, respectively. For the purpose of this example we assume that they do not change and that is why we do not specify the superindex. Alice, who likes keep her weekends private can write the following policy* $\forall i\colon Ag \cdot \forall l\colon Locs \cdot \forall t \cdot weekend(t) \implies (\forall t' \cdot t' > t \implies K^{t'}_i picture^t(Alice, l))$*. In English it means "Nobody can know Alice location during the weekend".*

VII.3.3 Properties of the framework

In this section we present interesting properties of our framework, together with a set of novel derived operators not present in traditional epistemic logics.

[2]For readability we omit $occurred^{15}(enter(B^{15}_{Alice} loc^{15}(Bob, work)))$ in Fig. VII.3 which is included in $EKB^{\sigma[15]}_{Alice}$.

To Learn or not to learn; To believe or not to believe. In $\mathcal{KBL}_\mathcal{T}$ we introduced the learning modality $L_i\varphi$ [69]. It stands for i learnt φ at a moment t when the formula is being evaluated. Here $L_i\varphi$, or more precisely, $L_i^t\varphi$ becomes a derived operator.

We say that an agent has learnt φ at time t, if she knows φ at time t but she did not know it for any previous timestamp. Formally, we define it in terms of $K_i^t\varphi$ as follows

$$L_i^t\varphi \triangleq \neg K_i^{pred(t)}\varphi \wedge K_i^t\varphi.$$

We can also model when users start to believe something, or *accept* a belief. To model this concept we define the acceptance operator. As before, it can be expressed using B_i^t

$$A_i^t\varphi \triangleq \neg B_i^{pred(t)}\varphi \wedge B_i^t\varphi.$$

Analogously we can express when users *forget* some knowledge or when they *reject* a belief. Intuitively, an agent forgets φ at time t if she knew it in the previous timestamp, i.e., *pred*(t)—recall the definition of *pred* in Section VII.3.2—and in t she does not know φ, and, analogously, for reject. Formally,

$$F_i^t\varphi \triangleq K_i^{pred(t)}\varphi \wedge \neg K_i^t\varphi$$

$$R_i^t\varphi \triangleq B_i^{pred(t)}\varphi \wedge \neg B_i^t\varphi.$$

Temporal modalities. The traditional temporal modalities $\Box$ and $\Diamond$ can easily be defined using quantification over timestamps as follows:

$$\Box\varphi(t) \triangleq \forall t \cdot \varphi(t)$$

$$\Diamond\varphi(t) \triangleq \exists t \cdot \varphi(t)$$

where $\varphi(t)$ is a formula φ which depends on t.

How long do agents remember?. Agents remember according to the length of the parameter ω. It can be seen as the size of their memory. Intuitively, increasing agents memory could only increase her knowledge. We prove this property in the following lemma.

Lemma 1 (Increasing window and Knowledge)**.** *Given σ, $t \in \sigma$ and $w, w' \in \mathbb{N}$ where $w \leq w'$.*

If $EKB_i^{\sigma[t]} \vdash (K_i^t\varphi, w)$, then $EKB_i^{\sigma[t]} \vdash (K_i^t\varphi, w')$.

Proof. Assume that $EKB_i^{\sigma[t]} \vdash (K_i^t \varphi, w)$. By Definition 6, there exists a derivation $(\varphi_1, w_1)\ (\varphi_2, w_2) \ldots (\varphi_n, w_n)$ for $n \in \mathbb{N}$ such that $(\varphi_n, w_n) = (\varphi, w)$. Let $\alpha = w' - w$, since $w \leq w'$ it follows that $\alpha \geq 0$.

Consider now the following derivation where the same deduction rules as in the previous derivation has been applied, and α is added to each w_i, $(\varphi_1, w_1 + \alpha)\ (\varphi_2, w_2 + \alpha) \ldots (\varphi_n, w_n + \alpha)$. We show now that, if $(\varphi_1, w_1)\ (\varphi_2, w_2)$ using a deduction rule R then $(\varphi_1, w_1 + \alpha)\ (\varphi_2, w_2 + \alpha)$ can also be derived using R, for all R in Table VII.2. We split the proof in derivation rules which, copy, reduce or introduce w.

- Rules that copy w. These are, A2, A3, A4, A5, K, B4, B5, L1 and L2. If $w \in \mathbb{N}$ given that $\alpha \geq 0$ it trivially follows that $w + \alpha \in \mathbb{N}$ which complies with the conditions of any of these rules. In this case $w = w'$ therefore the same applies to w'.

- Rules that reduce w. This is, KR1. In this case $w < w'$. In order for the derivation to be correct both w and w' are in $\mathbb{N}$. Since $\alpha \in \mathbb{N}$, it follows that $w + \alpha$ and $w' + \alpha$ are in $\mathbb{N}$. Moreover, since α is a constant it also follows that $w - w' = (w + \alpha) - (w' + \alpha)$. From the previous statement we conclude that the same window increase is required and, therefore the same derivation is performed.

- Rules that introduce w. These are, A1, D and Premise. No conditions are imposed in the value of w in order to apply these rules. Therefore, if they can be applied with window w since $\alpha \geq 0$ they can also be applied with window $w + \alpha$.

□

We can characterise how long agents remember information depending on the parameters ω and β of the framework. We differentiate for how long agents remember knowledge or beliefs since the parameter β might influence it.

Lemma 2 (ω knowledge monotonicity)**.** *Given σ and $t \in \mathbb{T}_\sigma$. If $K_i^t \varphi \in EKB_i^{\sigma[t]}$ then for all $t' \in \mathbb{T}_\sigma$ such that $t \leq t' \leq t + \omega$ it holds $\sigma \vDash K_i^{t'} \varphi$.* □

Proof. Assume $K_i^t \varphi \in EKB_i^{\sigma[t]}$. By premise we can derive $(K_i^t \varphi, 0)$. Let $t' = t + \omega$, by applying KR1 we can derive $(K_i^{t'} \varphi, \omega)$. By $\vDash$ we conclude $\sigma \vDash K_i^{t'} \varphi$. Given the above and by Lemma 1, for all $t \leq t'' < t + \omega$ it always possible to derive $(K_i^{t''} \varphi, \omega)$. □

This parameter give us a lot of flexibility in modelling agents memories. By choosing $\omega = \infty$ we can model agents with *perfect recall*, i.e, agents that never forget. Or we can also $\omega = 0$, i.e., agents who do not remember anything.

When $\beta =$ *conservative* memories about beliefs behave in the same way as knowledge.

Lemma 3 (ω conservative belief monotonicity). *Given σ and $t \in \mathbb{T}_\sigma$. If $B_i^t\varphi \in EKB_i^{\sigma[t]}$ and $B_i^{pred(t)}\varphi \notin EKB_i^{\sigma[t]}$, then for all $t' \in \mathbb{T}_\sigma$ such that $t \leq t' \leq t + \omega$ the following holds $\sigma \vDash \neg K_i^{t'}\neg\varphi \implies B_i^{t'}\varphi$.* □

Proof. The condition $B_i^t\varphi \in EKB_i^{\sigma[t]}$ and $B_i^{pred(t)}\varphi \notin EKB_i^{\sigma[t]}$ ensures that $B_i^{pred(t)}\varphi$ was not propagated. By definition Ψ_c contains the beliefs that are to be propagated from *pred*(t) to t up to the parameter ω. There three cases in which the EKB can be updated: i) No new information enters EKB; ii) New knowledge enters EKB; iii) A new belief enter the EKB. We show that the lemma holds for the three previous cases.

i) It trivially follows by the definition of belief propagation and Ψ_c.

ii) Assume $\sigma \vDash \neg K_i^{t'}\neg\varphi$ for all $t \leq t' \leq t + \omega$. Then for any new knowledge ψ that enters i's EKB it holds $\psi \neq \neg\varphi$, otherwise it holds $\sigma \vDash K_i^{t'}\neg\varphi$ thus deriving a contradiction. Hence, by definition of belief propagation it follows that $B_i^{t'}\varphi \in \Psi_c$ for all $t \leq t' \leq t + \omega$ and, consequently, $\sigma \vDash \neg K_i^{t'}\neg\varphi \implies B_i^t\varphi$.

iii) Let $B_i^{t'}\psi$ the belief to be introduced for any $t \leq t' \leq t+\omega$. By the definition of belief propagation, if $EKB_i^{\sigma[pred(t')]} \cup \Psi_c \cup \{B_i^{t'}\psi\} \vdash B_i^{t'}\neg\varphi$ then $EKB_i^{t'} = EKB_i^{pred(t')} \cup \Psi_c$ and, since $B_i^{t'}\psi \notin \Psi_c$ and $B_i^{t'}\psi \notin EKB_i^{pred(t')}$, it follows $B_i^{t'}\psi \notin EKB_i^{t'}$. Hence, $B_i^{t'}\varphi$ is propagated. Therefore, it holds that $\sigma \vDash B_i^{t'}\varphi$. Otherwise, $EKB_i^{t'} = EKB_i^{pred(t')} \cup \Psi_c \cup \{B_i^{t'}\psi\}$, since $B_i^{t'}\varphi \in \Psi_c$ it holds that $\sigma \vDash B_i^{t'}\varphi$.

□

On the contrary, susceptible agents can reject a belief when exposed to new contradictory beliefs. Therefore, the duration of their beliefs can be limited by an event introducing new beliefs in the EKBs.

Lemma 4 (ω susceptible belief monotonicity). *Given σ and $t, t' \in \mathbb{T}_\sigma$ such that $t < t'$ and $t' - t \leq \omega$. If $B_i^t\varphi \in EKB_i^{\sigma[t]}$, $B_i^{t'}\neg\varphi \in EKB_i^{\sigma[t']}$, $B_i^{pred(t)}\varphi \notin EKB_i^{\sigma[pred(t)]}$ and $B_i^{pred(t')}\neg\varphi \notin EKB_i^{\sigma[pred(t')]}$, then for all $t'' \in \mathbb{T}_\sigma$ such that $t \leq t'' \leq t'$ it holds $\sigma \vDash \neg K_i^{t''}\neg\varphi \implies B_i^{t''}\varphi$.* □

Proof. The proof of Lemma 3 can easily be adapted for this case by considering only all timestamps t'' such that $t \leq t'' \leq t' \leq t + \omega$. □

Refined versions of the β can be considered. Here we only studied the two extreme approaches. It is also possible to consider different ω for different pieces of information. In any case, the results of the previous lemmas are general enough to capture these modifications of the framework.

$$\begin{array}{lll} \sigma \vDash_C \forall x.\delta & \text{iff} & \text{for all } v \in D_o,\ \sigma \vDash_C \delta[v/x] \\ \sigma \vDash_C [\![\neg\alpha]\!]_i^s & \text{iff} & \sigma \vDash \neg\alpha \\ \sigma \vDash_C [\![\varphi \implies \neg\alpha]\!]_i^s & \text{iff} & \sigma \vDash \varphi \implies \neg\alpha \end{array}$$

Figure VII.4: Conformance relation for $\mathcal{PPL}_{\mathcal{RT}}$

VII.4 Writing Privacy Policies

In this section we provide a language for writing privacy polices, $\mathcal{PPL}_{\mathcal{RT}}$. In a nutshell $\mathcal{PPL}_{\mathcal{RT}}$ is a restricted version of $\mathcal{KBL}_{\mathcal{RT}}$ wrapped with $[\![\ ]\!]_i^s$ to indicate the owner of the policy i and its starting time s.

Definition 8 (Syntax of $\mathcal{PPL}_{\mathcal{RT}}$)**.** *Given agents $a, b \in Ag$, a nonempty set of agents $G \subseteq Ag$, timestamps s, t, a variable x, relation symbols $c_m^t(a,b), a_n^t(a,b)$, $p^t(\vec{s})$, and a formula $\varphi \in \mathcal{F}_{\mathcal{KBL}_{\mathcal{RT}}}$, the* syntax of the real-time privacy policy language $\mathcal{PPL}_{\mathcal{RT}}$ *is inductively defined as:*

$$\begin{array}{lll} \delta & ::= & \delta \wedge \delta \mid \forall x.\delta \mid [\![\neg\alpha]\!]_i^s \mid [\![\varphi \implies \neg\alpha]\!]_i^s \\ \alpha & ::= & \alpha \wedge \alpha \mid \forall x.\alpha \mid \exists x.\alpha \mid \psi \mid \gamma' \\ \psi & ::= & c_m^t(a,b) \mid a_n^t(a,b) \mid \mathit{occurred}^t(e) \\ \gamma' & ::= & K_i^t\gamma \mid B_i^t\gamma \\ \gamma & ::= & \gamma \wedge \gamma \mid \neg\gamma \mid p^t(\vec{s}) \mid \gamma' \mid \psi \mid \forall x.\gamma \end{array}$$

We will use $\mathcal{F}_{\mathcal{PPL}_{\mathcal{RT}}}$ to denote the set of all privacy policies created according to the previous definition. To determine whether a policy is violated in an evolving social network, we formalise the notion of conformance for $\mathcal{PPL}_{\mathcal{RT}}$.

Definition 9 (Conformance Relation)**.** *Given a well-formed trace $\sigma \in TCS$, a variable x, a timestamp s, and an agent $i \in Ag$, the* conformance relation $\vDash_C$ *is defined as shown in Fig. VII.4.*

The definition is quite simple, especially compared to that of conformance of $\mathcal{PPL}_{\mathcal{T}}$ [69]. If the policy is quantified, we substitute in the usual way. The main body of the policy in double brackets is dealt with by simply delegating to the satisfaction relation.

Examples

Example 5. *Assume Alice decides to hide all her weekend locations from her supervisor Bob. She has a number of options how to achieve this, depending on what the precise meaning of the policy should be.*

If the idea she has is to restrict Bob learning her weekend location directly when she posts it, she can define

$$\delta_1 = \forall t \cdot [\![weekend(t) \implies \neg K^{t}_{Bob} location(Alice, t)]\!]^{\textit{2016-04-16}}_{Alice}$$

where the weekend predicate is true if the timestamp supplied represents a time during a weekend. This policy can be read as "if x is a time instant during a weekend, then Bob is not allowed to learn at x Alice's location from time x".

This, however, is a very specialized scenario that captures only a small number of situations. Bob is, for example, free to learn Alice's location at any point not during the weekend, or at any point during the weekend when Alice's location is no longer up-to-date. Though there might be scenarios where this might be the desired behavior, we can define a policy that seems much closer to the intuitive meaning of learning someone's location on a weekend. Consider

$$\delta_2 = \forall t \cdot [\![weekend(t) \implies \neg \exists t' \cdot (K^{t'}_{Bob} location(Alice, t))]\!]^{\textit{2016-04-16}}_{Alice}.$$

Here, Bob is not allowed to learn Alice's location from a weekend, no matter when. If the policy does not get violated, then Alice's weekend locations will be completely safe from Bob – on the social network, at least.

Example 6. *One of the advantages of $\mathcal{PPL}_{\mathcal{RT}}$ is the ability to distinguish between the original time of a piece of information and the time when it should be hidden. Suppose Diane activates the following policy:*

$$\delta_1 = \forall t \cdot \forall x \colon Ag^t \cdot [\![\neg friends^t(Diane, x) \implies \neg \exists t' \cdot (K^{t}_{x} post^{t'}(Diane))]\!]^{\textit{2016-05-28}}_{Diane}$$

This aims to prevent anyone who is not a friend of Diane's from learning any of her posts (here we assume that the friends connection is not reflexive for simplicity, otherwise the restriction would target Diane herself, too).

Though δ_1 may seem reasonable enough, it might be unnecessarily restrictive. Let us say there is another user, Ethan. Diane becomes friends with Ethan on May 31, so when her policy is already in effect. Should Ethan be able to learn about Diane's posts from when they were not friends? Not according to δ_1, which says that no one, regardless of their relationship with Diane at the moment, is able to learn about her posts from when they were not friends.

Note that while this may indeed be the desired behaviour, it is, for example, not what happens on Facebook, where when two users become friends, they are free to access each other's timeline including past events and posts. $\mathcal{PPL}_{\mathcal{RT}}$ is expressive enough to model *this behaviour as well. We can define:*

$$\delta_2 = \forall t \cdot \forall t' \cdot \forall x\colon\, Ag^{t}[\![\neg \mathit{friends}^{t}(\mathit{Diane}, x) \wedge \neg \mathit{friends}^{t'}(\mathit{Diane}, x) \implies \neg K_x^{t'} \mathit{post}^{t}(\mathit{Diane})]\!]_{\mathit{Diane}}^{\mathit{2016\text{-}05\text{-}28}}$$

This policy precisely defines the point in time from *when to hide information, y, as well as the point in time* when *to hide it, y′. It says, "if Diane is not friend with someone, then that someone cannot learn her posts, but only if they come from a time when they were not friends". Note that δ_1 says nothing about users who are currently friends of Diane's, which is different from δ_1 – here her friends can learn anything, including past posts from when they were not friends with her.*

VII.5 Related work

Specifying and reasoning about temporal properties in multi-agent systems using epistemic logic have been previously studied in a number of papers (e.g., in [29] for *interpreted systems*; see also [69] and references therein). More recently, Moses *et al.* have extended interpreted systems to enhance reasoning about past and future knowledge. In [8] they extend K_i with a time-stamp $K_{i,t}$, allowing for the expression of properties such as "Alice knows at time 10 that Bob knew p at time 1", i.e., $K_{Alice,10} K_{Bob,1}\ p$. Thought there are some similarities between the work by Moses *et al.* and ours, namely the use of time-stamps in the knowledge modality and the *moving* along a trace to place the evaluation in the "right" place, there are quite a few differences. First, we differ in the intended use of the logics: Moses *et al.* use time to model delays in protocols, whereas our main motivation is to provide a rich privacy policy language for OSNs. Besides, our logic includes belief and other operators not present in the timed versions introduced by them, and we have time-stamps associated with propositions. We claim, however, that $\mathcal{KBL}_{\mathcal{RT}}$ as at least as expressive as the logics introduced by Moses *et al.* , though this would need to be formally proved and for that we would need to relate our (non-standard) semantics with interpreted systems. This is left as future work.

As mentioned in the introduction, our work solves the open issues and limitations described in our previous work [69]. In that paper we introduced $\mathcal{PPF}_{\mathcal{T}}$, a temporal epistemic framework for describing policies for OSNs. $\mathcal{PPF}_{\mathcal{T}}$ relies on the temporal epistemic logic $\mathcal{KBL}_{\mathcal{T}}$ that allows to express temporal constraints using the classical *box* and *diamond* temporal operators. Neither the policy language nor the underlying logic $\mathcal{KBL}_{\mathcal{T}}$ have a notion of time: this is only used at the semantic level, allowing to move along a social network model trace in order to interpret the temporal operators. $\mathcal{PPF}_{\mathcal{RT}}$ strictly extends $\mathcal{PPF}_{\mathcal{T}}$, so our work not only addressed the already identified limitations identified in [69] but also extends that work by allowing the definition of more modalities (e.g., *forget*, *accept*, *belief*) allowing for a more expressive policy language and underlying logic. These allow us to define policies for more complex OSNs, like Snapchat.

Besides the above, it is worth mentioning the work by Woźna & Lomuscio [97] where TCTLKD s presented. TCTLKD is a combination of epistemic logic, CTL, a deontic modality and real time. It is difficult to compare our logic $\mathcal{KBL}_{\mathcal{RT}}$ with TCTLKD as they use CTL, while we have time-stamps in the propositions. The models used to interpret formulae in TCTLKD are based on a semantics for a branching logic, being a combination of timed automata and interpreted systems plus an equivalence relation for modelling permission. Ours is based on (timed) social network models. Besides we can also reason about belief.

VII.6 Conclusions

In this paper, we have presented a novel privacy policy framework based on a logic that offers explicitly support for expressing timestamps in events and epistemic operators. This framework extends [66, 71], which did not offer any support for time, and [69] which only had limited support due to the implicit treatment of time. Our framework is based on Extended Knowledge Bases (EKB). A query to an EKB starts by instantiating a number of epistemic axioms that handle knowledge, belief and time (the concrete axioms depend on the OSN instantiation). The deductive proof system give an algorithm to deduce the knowledge of agents acquired at each instant, and in turn a model checking algorithm for the logic and a check for privacy policy violations. The explicit time-stamps allow to define learning and forget operators that capture when knowledge is acquired. Similarly, one can derive accept and reject operators that model when beliefs come into existence and are rejected.

The flexibility of the EKBs allows to model different kinds of OSNs in terms of how the actions affect knowledge and how this knowledge is preserved through time. We have also sketched how different existing OSNs can be modeled using this flexibility.

Two important avenues for future research are the following. First, many instantiations enable efficient implementations of checking privacy policy violations by exploiting whether events can affect the knowledge of the agents involved. Once the effect of the actions is fixed one can prove that a distributed algorithm guarantees the same outcome as the centralized algorithm proposed here. For example, tweets can only affect the knowledge of subscribers so all other users are unaffected. Second, once an effective system to check policy violations is in place, there are different possibilities that the OSN can offer. One is to enforce the policy by forbidding the action that the last agent executed, which would lead to the violation. Another can be the analysis of the trace to assign blame (and correspondingly, reputation) to the agents involved in the chain of actions. For example, the creator of a gossip or fake news may be held more responsible than users forwarding them. Even a finer analysis of controllability can give more powerful algorithms by detecting which agents could have prevented the information flow that lead to the violation. Finally, yet another possibility would be to remove past events from the history trace of the OSN creating a pruned trace with no violation. All these possibilities are enabled by having a formal framework like the one presented in this paper.

Chapter VIII

Secure Photo Sharing in Social Networks

Pablo Picazo-Sanchez, Raúl Pardo and Gerardo Schneider

Abstract. Nowadays, in an ubiquitous world where everything is connected to the Internet and where social networks play an important role in our lives, security and privacy is a must. Billions of pictures are uploaded daily to social networks and, with them, parts of our private life are disclosed. In this work, we propose a practical solution for secure photo sharing on social network with independence of its architecture which can be either centralised or distributed. This solution solves the inconsistencies that appear in distributed social network as a consequence of treating photos and access policies separately. Specifically, we solve this open problem by attaching an access policy to the images and thus, each time a photo is re-shared, the access policy will travel together with the image.

VIII.1 Introduction

Online Social Networks (OSNs) such as Facebook, Twitter or Instagram are only a few examples of the most used Internet applications all over the world. A recent study shows that Facebook [88] has at least 1.71 billion active users per month. Moreover, according to that study, it is estimated than more than 300 million photos per day are being uploaded.

Most OSN users have the tendency to share photos. There are several works that are focused on the reason for sharing personal information such as photos on OSNs from a sociological perspective [24, 50, 52, 58]. These studies found out that most users share photos on OSNs to seek affection. Nevertheless, users are aware of the risks of their actions which might reveal personal aspects of their lives. Due to this, users usually weight the risks of disclosing private information against benefits of not doing it.

Both security and privacy issues have been pointed out in several papers as unsolved and challenging problems [48]. Specifically, in the privacy domain, some authors have addressed *photo sharing*[1] as an open problem in OSN [90, 48].

This problem arises when users take photos they have access to and increase the audience of the photo by re-sharing it. For instance, imagine that Alice shares a photo with her friends, and later, Bob—who is a friend with Alice—re-shares it with his own friends, thus increasing the audience to his own friends as well. Essentially, this circumstance is given because the privacy policies that Alice has previously defined are applied only to her public domain and are not attached to the objects she shares out.

OSNs can be classified into centralised and distributed social networks. In centralised OSNs there is only one instance which has a global view of the state of the system and where all information is handled. On the other hand, in *Distributed Online Social Networks (DOSNs)*, there are different servers where each one of them has its own instance of the OSN and has the ability of sharing and exchanging information between them.

Facebook, Twitter or Instagram are some examples of centralised OSNs. However, under the hood, the store infrastructure of these OSNs is geographically distributed. For instance, Facebook developers have deployed a distributed data store for the resources of the OSN [11, 65]. This storage system is based on a master/slave architecture which replicates the information geographically so that it is accessed efficiently. Bronson *et al.* pointed out in [11] that their storage system explicitly favours availability and per-machine efficiency over strong consistency. They also remarked the problem of *expensive read-after-write consistency*, i.e., the cost of forwarding writes to the master and later being replicated, and the existence of time elapses before all slaves have a

[1]It is also known as *photo re-sharing* since photos can be shared many times and by different users.

consistent information. In the context of photo sharing, it might originate problems while updating the audience of a photo. Imagine that Alice initially shares a photo with her friends, but after a while she decides to restrict the audience to her family and rewrites the access control policy of the photo. Before this policy is replicated in the whole system—a few milliseconds according to [11]—there will be slaves which would show Alice's photo to the incorrect audience.

Diaspora [22] is the most popular example of DOSNs with more than 0.6 million users. Moreover, in Diaspora, each server is called a *pod* and has its own database. Thus, this architecture prevents a single party to have all the users' personal information. In a DOSN when users from different nodes of the system share information, it is replicated on each node. This highly distributed architecture makes very hard to keep consistency between pods and it directly affects the photo sharing problem we are tackling here. Furthermore, in Diaspora after a user has shared a photo, it is not possible to update its access control policies. This is because once the photo is replicated, a static access control policy is sent to specify the audience of the photo in that pod. Due to this unpleasant restriction, inconsistencies when a user updates the relationships with users from different pods may appear. For instance, imagine that Alice shares a photo with her friends. Bob, who signed up in a different pod, gets access to the photo, given that it was replicated to his pod and the access control policy allows him to see it. A few days afterwards, Alice decides to end her friendship with Bob. One would expect Bob to not be able to see the photo shared with Alice's friends. However, the unfriend event is not replicated to all pods where the photo was sent, and therefore Bob continues having access to the photo.

Note that in both architectures the problem arises from having two separate entities, i.e., the photo and its access control policy, and inconsistencies while updating the access control policy of a photo. Here we propose a solution where access control policies are "stuck" to the photo. Therefore when a photo is replicated in different nodes, its access policy travels together with it.

Contributions. We focus on how to share private images on DOSN in a secure way. To do so, we have developed a solution where the access policy is attached to the image by using *Attribute Based Encryption (ABE)*, instead of defining a common access control policy in the generic privacy settings, e.g., "only family" or "colleagues and friends". Moreover, we have tested our proposal on Diaspora to demonstrate its viability on both modes centralised and decentralised[2]. As far as we know, this is the first solution which allows different images formats such as PNG, JPEG or TIFF. Finally, by using the centralised mode of Diaspora, we show how this could be easily

[2]Accessible online at `http://ppf-diaspora.raulpardo.org`

deployed into real applications such as Facebook, Twitter or any other OSN.

The rest of this paper is organised as follows: Section VIII.2 introduces background knowledge on ABE. In Section VIII.3 we present our system design and the core of our proposal. Section VIII.4 presents the results and the experiments we have run. In Section VIII.5 we give an overview of works on OSNs from the security and privacy photo re-sharing point of view, and present a comparison with our approach. We conclude and describe future work in the last section.

VIII.2 Preliminaries

For completeness and readability, this section provides a brief overview of the cryptographic primitives and security assumptions used throughout the paper.

VIII.2.1 Access Structure

Let $\mathcal{U}$ be the attribute universe and $\mathbb{A}$ a non-empty collection of attributes $\{Att_1, Att_2, \dots, Att_n\}$, with $Att_i \in \{0,1\}^n$. $\mathbb{A}$ is an access structure over $\mathcal{U}$ where the sets specified by $\mathbb{A}$ are called the authorised sets. Notice that each time that new users join the network, a set of attributes is assigned to them.

Moreover, an access structure $\mathbb{A} \subseteq \mathcal{U}$ is monotone if $\forall B, C \subseteq \mathcal{U}$ if $B \subseteq \mathbb{A}$ and $B \subseteq C$ then $C \subseteq \mathbb{A}$.

VIII.2.2 Linear Secret Sharing Scheme

Informally, a *secret-sharing scheme* among a dealer and a set of parties is an algorithm in which a secret k is distributed to a set of i parties in such way that only authorised subsets of parties can reconstruct the secret by pooling the shares of the authorised parties, while unauthorised subsets will learn nothing about the secret. Additionally, when the secret is a random vector chosen over $\mathbb{Z}_p$ is called *linear* secret sharing scheme.

Furthermore, we assume that when an access structure $\mathbb{A}$ is given as a monotonic boolean formula over a set of attributes, there is a polynomial time algorithm that translates it to the matrix access policy [46]. Formally, let p be a prime number and $\mathcal{U}$ the attribute universe, a secret-sharing scheme Π with domain of secrets $\mathbb{Z}_p$ realising access structures on $\mathcal{U}$ is *linear* over $\mathbb{Z}_p$ if:

- The shares of a secret $k \in \mathbb{Z}_p$ for each attribute form a vector over Z_p;
- There exists an $l \times n$ matrix $M \in \mathbb{Z}^{l \times n}$, called the *share-generating matrix*, where for all $x = 1, \dots, l$, the x-*th* row of M is labelled by a function $\rho(x)$ (from $\{1, \cdots, l\}$ to $\mathcal{U}$). Additionally, during the shares generation, if we consider the column vector $v = (k, r_2, \dots, r_n)^t$, where $r_2, \dots, r_n \in \mathbb{Z}_p$ are randomly chosen, then the vector

of l shares of the secret k according to the Π is $Mv \in \mathbb{Z}_p^{l\times 1}$. The share $(Mv)_x$ belongs to $\rho(x)$.

VIII.2.3 Multi-Authority Attributes

Since our solution uses the *Multi Authority-Attribute Based Encryption (MA-ABE)* scheme proposed in [79], we do assume that there is a computable function T which links each attribute $\mathcal{U}$ to a unique authority φ of the set of authorities $\mathcal{U}_\varphi$ i.e., $\mathrm{T} : \mathcal{U} \rightarrow \mathcal{U}_\varphi$. Moreover, this function creates a second labelling of rows in the policy $(\mathbb{A},\rho)$, which maps rows to attributes by $\mathrm{T}(\rho(x))$. We additionally follow the same notation introduced in the original paper where the attributes are defined according to the next pattern: [attribute-id]@[authority-id].

VIII.2.4 Bilinear Pairings

Informally, a pairing function is a function that associates each pair of values of a given set with a single value of the set. A bilinear parting function is a pairing function that satisfy bilinear, non-degenerate, efficient and symmetric properties. More formally, let $\mathbb{G}$ and $\mathbb{G}_T$ be two multiplicative cyclic groups of the same prime order p, g a generator of $\mathbb{G}$, and $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ a pairing function satisfying the following properties:

- Bilinear: $\forall u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$; we have $e(u^a, v^b) = e(u, v)^{ab}$.
- Non-degenerate: $e(g, g) \neq 1$, i.e., the identity element of $\mathbb{G}_T$.
- Efficient: there is an efficient algorithm to compute $e(u, v), \forall u, v \in \mathbb{G}$.
- Symmetric: e is symmetric, i.e., $e(g^a, g^b) = e(g, g)^{ab} = e(g^b, g^a)$.

It is important to mention that both authorities and users are provided with a unique identifier *GID* which is mapped by a function $\mathcal{H}$ to an element in the group $\mathbb{G}$, i.e., $\mathcal{H} : \mathit{GID} \rightarrow \mathbb{G}$. Additionally, we define another function $\mathcal{F}$ that translates attributes to elements in a group $\mathbb{G}$, i.e., $\mathcal{F} : \mathit{Att} \rightarrow \mathbb{G}$.

VIII.2.5 Security Assumptions

Similarly to [79], the security of our proposal relies on the *q-type* assumption (q-DPBDHE2 in short) which basically is a slight modification of the q-Decisional Parallel Bilinear Diffie-Hellman Exponent Assumption [95]. The following definition has been previously demonstrated in [95], so we encourage the reader to check the full security proof.

Let $a, s, b_1, \cdots, b_n \in \mathbb{Z}_p$ be randomly chosen and g a generator of $\mathbb{G}$ of prime order p. If an adversary $\mathcal{A}$ is provided with $\{\mathbb{G}, p, e, g, g^s\} \cup D$ where D is:

$$D = \left(\left\{ g^{a^i} \right\}_{\substack{i \in [2q] \\ i \neq q+1}}, \left\{ g^{a^i b_j} \right\}_{\substack{(i,j) \in [2q,q] \\ i \neq q+1}}, \left\{ g^{s/b_i} \right\}_{i \in [q]}, \left\{ g^{s a^i b_j / b_{j'}} \right\}_{\substack{(i,j,j') \in [q+1,q,q] \\ j \neq j'}} \right)$$

for any probabilistic algorithm $\mathcal{B}$, the advantage of $\mathcal{A}$ in solving the q-DPBDHE2 problem is negligible i.e., this assumption relies on the fact that the probability of distinguishing the bilinear pairing $e(g,g)^{sa^{q+1}}$ from a random element $R \in \mathbb{G}_T$ is negligible:

$$Adv_{\mathcal{B}}^{q-DPBDHE2} = \left| Pr\left[\mathcal{B}(D, e(g,g)^{sa^{q+1}}) = 0 \right] - Pr\left[\mathcal{B}(D,R) = 0 \right] \right| \leq \epsilon$$

VIII.2.6 MA-ABE Algorithms

The MA-ABE scheme is mainly based on four different algorithms: *GlobalSetup*, *AuthSetup*, *KeyGen*, *Encrypt* and *Decrypt*. In the following we summarise the five algorithms (for a more detailed description check [79]):

- *GlobalSetup*$(1^\lambda) \rightarrow GP$. This method requires a security parameter λ. It outputs the global parameters $GP = \{p, \mathbb{G}, g, \mathcal{H}, \mathcal{F}, \mathcal{U}, \mathcal{U}_\varphi\}$.
- *AuthSetup*$(GP,\ \varphi) \rightarrow \{PK_\varphi, SK_\varphi\}$. This algorithm generates both a public and a private key for each one of the authorities.
- *KeyGen*$(GID,\ \varphi,\ Att,\ SK_\varphi,\ GP) \rightarrow SK_{GID,Att}$. This method takes as input the user's *GID*, the authority φ, the attribute *Att*, the secret key of the authority SK_φ and the general parameters *GP* and it outputs the user's secret key for a given attribute *Att* —controlled for the authority φ.
- *Encrypt*$(\mathrm{M},\ \mathcal{T},\ \{PK_\varphi\},\ GP) \rightarrow CT$. This algorithm is run by the users and it receives as input the message to be encrypted M, the access policy $\mathcal{T} = (\mathbb{A}, \rho)$, the public keys of the authorities $\{PK_\varphi\}$, and the general parameters GP. It outputs the ciphertext CT (ciphered under the access policy $\mathcal{T}$) together with $\mathcal{T}$.
- *Decrypt*$(CT,\ \{SK_{GID,Att}\},\ GP) \rightarrow M$. When a user wants to decrypt a ciphertext, she runs this algorithm. The GP, the ciphertext CT and all the secret keys of that user $SK_{GID,Att}$ (to recover the shares of the access matrix) should be provided to get the plaintext.

VIII.3 System Design

In this section we explain in detail our proposed solution for re-sharing photos in DOSNs. Concretely, we describe the design we implemented in Diaspora.

VIII.3.1 Diaspora's Architecture and Assumptions

As mentioned in the introduction, Diaspora is a very popular DOSNs. The source of its popularity lies on a distributed architecture which prevents a single party to control users' data. Moreover, Diaspora can work as a centralised social network if there is only one pod in the system.

The distributed architecture of Diaspora consists of *pods*. A *pod* is a server which runs an instance of Diaspora's source code. In order for users to join Diaspora they can either join an existing pod or create their own. Every pod has its own database, therefore when users join a pod, their information is not available to everyone. Moreover, only the owner of the pod has direct access to the information of the database.

Users can connect with other users from the pod they joined as well as users who signed up in other pods. As usual in OSNs, they can define connection relations to classify their contacts such as *friends*, *acquaintances*, *family* and so on. Using these relations, users can define the audience of their information, i.e., posts, photos, polls, etc. When information is shared with users from different pods it needs to be replicated. For example, when a set of photos are accessed in different pods then they are replicated in the databases of each one of the involved pods. After the photo is replicated, the access control policies (of the target pod) are updated to determine which users in the pod can access it. If the owner of the pod were to update the photo audience, the access control policies should be updated in all the pods where the photo was distributed to.

Note that this approach requires distributing the photo and (separately) the access control policy. In this way, consistency errors can easily appear, e.g., if the photo is successfully distributed but there is an error while distributing the access control policy. An additional problem is updating the policies of a photo. If a user decides to update the audience of a photo from her friends to nobody, this policy must be transmitted to all the pods where the photo has been replicated. As before, it can originate inconsistencies, for instance, when a pod with a replica of the photo loses connectivity. Currently in Diaspora it is not possible to update the access control policies of a photo after sharing it. This is, probably, because of the difficulties to enforce consistency in such a distributed environment. The previous example can be seen in Figure VIII.1.

Finally, in our proposal we assume the following: i) the *KeyGen* algorithm is only run by the pods of Diaspora and thus they are trustworthy; ii) photos can be stored either in the pods or in public repositories so it is not mandatory to be secure; and iii) there is a function named *getAtt* that given a user, it returns the set of a attributes of the user from all the pods in the network.

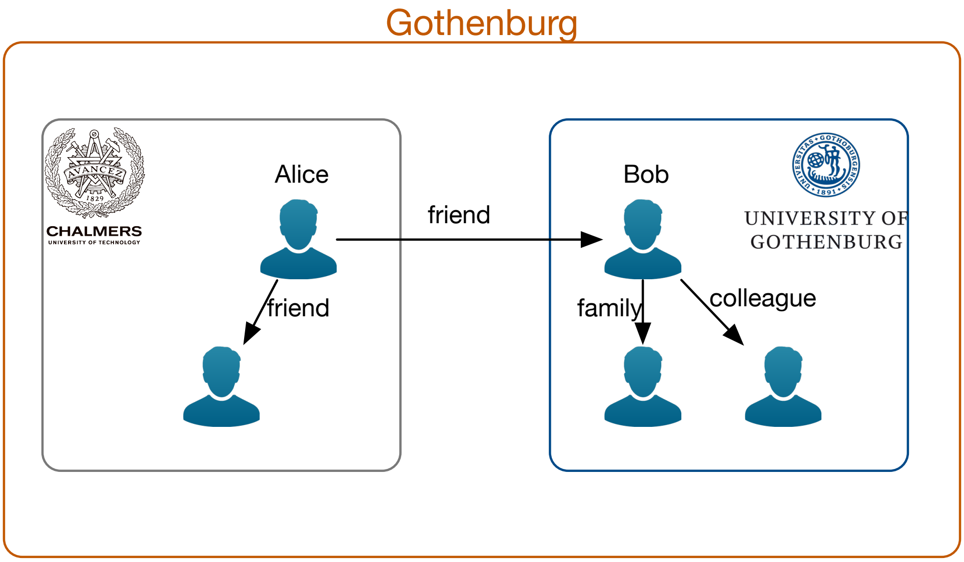


Figure VIII.1: DOSN example.

VIII.3.2 MA-ABE in Diaspora

In our solution we propose to attach the "access control policies" to the photo by using a decentralised version of ABE. Classical ABE approaches are based on a centralised assumption where a *Trusted Party (TP)* is in charge of distributing the keys of the scheme and sets up the system. However this is infeasible because of two main problems: 1) the TP has the power to decrypt everything in the system and 2) there is no practical solution if there are n-different authorities running the same cryptographic schema and users from different authorities want to share information with them.

In a nutshell, our approach consists in encrypting (parts of) the photo with a policy which specifies the attributes that other users must possess in order to see the encrypted parts. In what follows we provide a detailed description of our design of photo sharing in Diaspora based on MA-ABE.

Attributes in Diaspora. We define the attribute universe, $\mathcal{U}$, to be the set of all possible connections between users. For instance, in a pod with only two users, Alice and Bob, and the *friend* relation, the universe of attributes is $\mathcal{U} = \{\textit{friend}(\text{Alice}), \textit{friend}(\text{Bob})\}$. The attribute *friend*(Alice) will be granted to users that Alice marked as friends. In general, given a set of users $\mathcal{US}$ and a set of connections $\mathcal{C}$, the shape of $\mathcal{U}$ is as follows: $\mathcal{U} = \{c(u) \mid \forall u \in \mathcal{US}, \forall c \in \mathcal{C}\}$.

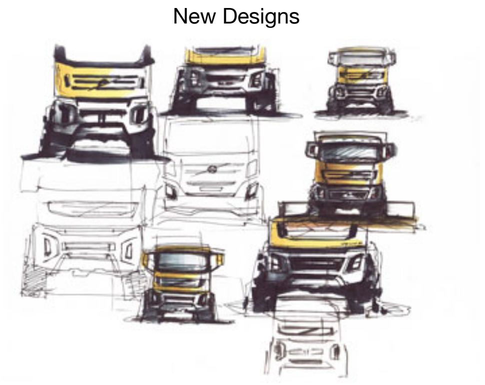
The universe of attributes in the system is not centralised. Due to Diaspora's distributed architecture, the universe of attributes is composed by the attributes in each pod. Let $\mathcal{U}_{Chalmers}$ and $\mathcal{U}_{GU}$ be the universe of attributes of the Diaspora pods of *Chalmers University* and the *University of Gothenburg (GU)*, respectively. We say that the universe of attributes in Gothenburg is $\mathcal{U}_{GBG} = \mathcal{U}_{Chalmers} \cup \mathcal{U}_{GU}$. We use the same notation to denote the set of users $\mathcal{US}_{GBG} = \mathcal{US}_{Chalmers} \cup \mathcal{US}_{GU}$ and the set of connections in Gothenburg pods $\mathcal{C}_{GBG} = \mathcal{C}_{Chalmers} \cup \mathcal{C}_{GU}$.

In this way, diaspora pods act as authorities which grant attributes to users. Determining whether a user has an attribute can be easily checked by querying the database of the pod. Note that users might have attributes which belong to different pods, e.g., Alice (from the Chalmers pod) can mark Bob (from the GU pod) as *friend*. Therefore, Bob will have attributes that come, not only from the GU pod, but also from the Chalmers pod. We use the same notation as in the original definition of MA-ABE in [79] to specify the provenance of an attribute, e.g., *friend*(Alice)@Chalmers. This example can be seen in Figure VIII.1.

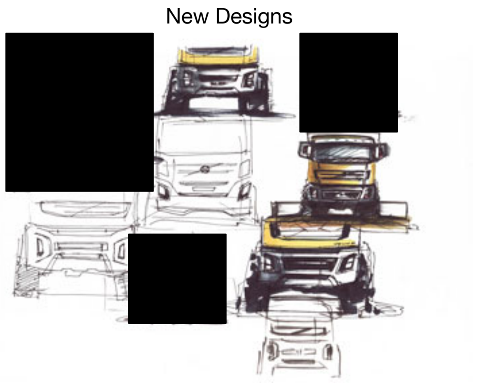
Key Generation. Initially, when users join Diaspora, they have no connection to other users. Thus, they possess no attributes. As they interact with the system they start to create new connections, and consequently, grant (and be granted with) new attributes. As we mentioned in the preliminaries section, there exists a *KeyGen* algorithm which given the attributes $Att_1, \ldots, Att_n$ of a user, her *GID* and some additional parameters, it produces the corresponding secret keys, $SK_{GID,Att_1} \ldots, SK_{GID,Att_n}$ for $n \in \mathbb{N}$. Nevertheless, note that the set of attributes that a user has is dynamic, i.e., it will change as users interact with each other. Therefore, a very important question to answer is: When should the key generation step be carried out?

We chose to perform the key generation algorithm only when the set of attributes of a user changes. Checking a change in the set of attributes of a user requires performing a broadcast call to all pods in the network. We use a function $getAtt : \mathcal{US} \rightarrow 2^{\mathcal{U}}$ which given a user, it returns the set of attributes posses by the user in any pod in the network. Afterwards, we execute *KeyGen* for the new attributes of the user—in the corresponding pod—and remove the keys from attributes that might have been revoked[3]. Though executing *getAtt* is not computationally expensive, it requires communication between pods and might introduce delays, therefore it is important to minimise its use. Having an updated set of attributes is only necessary when decrypting photos since the

[3]We discuss other approaches to attribute revocation proposed in the literature in Section VIII.5.



(a) Original



(b) Encrypted

Figure VIII.2: Sample photo with and without encrypted area.

set of attributes that a user has determines which parts of the photo that are visible.

Therefore, in order to reduce the overhead of this operation to the minimum, we only execute *getAtt*—and the corresponding calls to *KeyGen*—after receiving a set of photos to show. This occurs, for instance, every time users access their stream of posts, or whenever they access a particular photo. Encrypting a photo does not require these secrets key (see Section VIII.2). It only requires having access to the plain attributes the user will use for the policy. As mentioned earlier, this attributes are easily accessible by querying the database.

Attaching policies to photos. In the same way that users can now choose the audience of a photo, in our proposal users choose the attributes that other users must have in order to access a photo. Moreover, we let users grab the area of the photo that they want to protect and the actions that can be performed with the photo e.g., re-share, like, comment, etc. This information constitutes the access policy, $\mathcal{T}$. The photo to protect together with $\mathcal{T}$—and, as before, some additional parameters, see Section VIII.2—are the input parameters of the encrypt algorithm, which returns a ciphertext CT. This ciphertext is distributed in the system and it contains both the picture and the access policy.

Example 1. *Imagine that the department of vehicle's design from Chalmers decides to use Diaspora to share the photo shown in Figure VIII.2a. However, this photo contains some parts that are still pending of the patent's decision and the researchers only want their colleagues to see the final design. In our system, researchers can* *select the part of the photo—where some compromised design appears—and encrypt it with the attribute colleague(*$Department_{design}$*)@Chalmers. Later users with the attribute colleague(*$Department_{design}$*)@Chalmers will be able to decrypt the photo and see Figure VIII.2a and the remaining users will see Figure VIII.2b.*

Several access policies can be attached to a photo. The only restriction we impose is that encrypted areas cannot be re-encrypted. For instance, let Alice be an engineer working at the Swedish vehicle manufacturer *Ovlov*, and also collaborating with the department of vehicle's design at Chalmers. She decides that there are some parts of the image that the researchers at Chalmers shared (Figure VIII.2b) that are still visible but should only be accessible by *Ovlov* employees. In other words, some areas of Figure VIII.2b that were not encrypted by Chalmers researchers. Therefore, she decides to encrypt some of those parts and share the photo again. The resulting ciphertext will allow users with the attribute *colleague(*$Department_{design}$*)@Chalmers* to only see some parts of the photo, users with the attribute *employee(Ovlov)@Ovlov* to see others parts of the image, and users with both attributes to see the complete photo.

Implementation. We have implemented a prototype of this system in our own Diaspora pod[4]. This prototype allows users to select the area of any of photos they have access to, and encrypt it with the access policy "Only my friends can see this area"—we plan to offer users the possibility to choose among other policies, such as friends of friends, colleagues, family or any other connection they have defined. It is not required to be the owner of the photo to be able to encrypt it. For instance, Alice can share with her friends a photo together with Bob. Later Bob—who is part of the audience of the photo—will be able to select part of the area of the photo that he considers private and encrypt it so that only his friends can see it. As a result, the friends of Alice who are also friends with Bob will be able to see the whole photo. On the other hand, the friends of Alice who are not friends with Bob will only see the non-encrypted parts of the photo.

VIII.4 Evaluation

In this section we show different experiments that have been run in order to test our solution to demonstrate that it can be deployed in Diaspora and thus, the security of this DOSN would improve considerably. Additionally, our proposed solution is open source and can be downloaded online[5].

[4]`https://ppf-diaspora.raulpardo.org`

[5]`https://github.com/raulpardo/ppf-diaspora/tree/abe-photos`

We have run the simulations 10 times and we have computed the time average. Additionally, we have deployed the solution in a real scenario using the *Amazon Web Services (AWS)* architecture. All AWS instances are catalogued as *t2.xlarge* in such environment. The characteristics in term of hardware are: 4 virtual Intel Xenon CPU with 16GB of RAM with no *Elastic Block Store (EBS)* storage system. Regarding the software, all instances are running a x64 architecture under Ubuntu 12.04 *Operating System (OS)*. The generated JSON files of the systems are in average: 4Kb (users' secret keys); 401kb (ABE's global parameters); 490Kb (authorities' keys) and for the *CT* some samples—which depend on the size of the photo to encrypt—are shown in Figure VIII.4 (in the worst case, i.e., encrypting the whole area of the photo).

Figure VIII.3 shows how ABE behaves when different amount of attributes take place when both algorithms encryption and decryption are run over an entire image of 800×574 pixels. In Figure VIII.3a we have fixed the number of attributes in the policy to 3, i.e., $|\mathcal{T}| = 3$. On the other hand, in Figure VIII.3b we have fixed the number of attributes in the universe to 100, i.e.,$|\mathcal{U}| = 100$. From these plots, it is interesting to see that the number of attributes do not affect to the performance and thus, taking into account that we have run our experiments in the worst case (encrypting the whole image), all results under 2 seconds in the decryption algorithm can be considered as good results. Finally, we can conclude that our distributed solution for photo re-sharing will perform perfectly when the number of attributes in the policy $\mathcal{T}$ is no higher than 13 attributes.

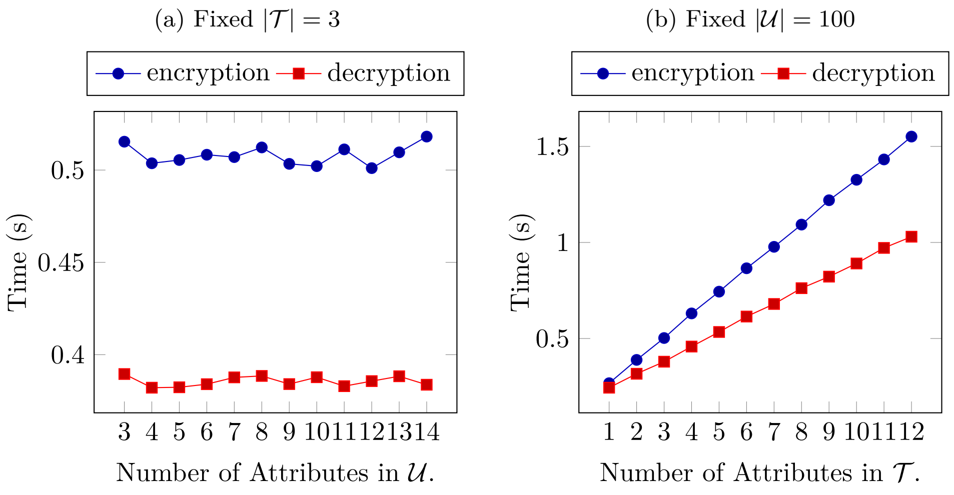


Figure VIII.3: Encryption and Decryption time in a 800x574 image.

We have run one more experiment to show how the size of the ciphertext CT is independent of both the numbers of attributes in the systems and the length of the access policy $\mathcal{T}$. However we have observed that the size of the CT generated is hardly dependent of both the photo's resolution and logically the selected area to be encrypted. In this experiment, we have used different images resolution and we have cyphered all the image –which rarely occurs– to be in the worst case. In this point, it is important to remark that Facebook re-sizes the images, and the widest side of image does not exceed 2048 pixels. In the Figure (VIII.4) can be seen that the generated CT depends on the resolution of the image. It was expected, because the larger the image is, the larger the area to cypher is. We have additionally tested if the size of the generated CT depends on either the number of attributes on the system $\mathcal{U}$ or on the number of attributes involved in the access policy $\mathcal{T}$ and we have realised that the size remains constant.

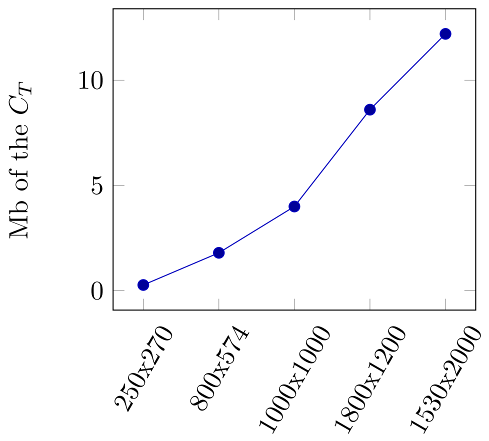


Figure VIII.4: C_T vs Resolution image.

VIII.5 Related Work

Despite the fact that there are several works that try to guarantee both security and privacy on photos, only few proposals specifically focus on DOSNs [4, 13, 19, 64, 77, 99, 101] and only a subset where ABE is used [4, 64, 99].

Nilizadeh *et al.* proposed a DOSN called Cachet [64] . The main characteristic of this schema is that both ABE and a symmetric encryption are used together. Basically, the secret key is encrypted using ABE and only those users that satisfy the policy will get the secret key and decrypt the content. This architecture is similar to the one proposed by Baden *et al.* some years before in [4].

Recently, a work published by Yuan *et al.* in [99] proposed to encrypt an image under an access policy by using an ABE scheme. This proposal uses three different encryption schemes: symmetric encryption, RSA and *Ciphertext Policy-Attribute Based Encryption (CP-ABE).* Symmetric encryption, in particular AES, is used to encrypt the areas of the image. The RSA algorithm is used to encrypt a secret key for a given user. Finally, CP-ABE is used to check who can access to a given secret key in order to decrypt a given photo.

ABE it is commonly used as an encryption scheme to share the secret key of a symmetric encryption such as *Advanced Encryption Standard (AES).* This is especially useful because symmetric encryption performance is significantly lower than any other public encryption schema. Additionally, by using this technique the size of the ciphertext produced by the ABE remains always constant.

However, using symmetric encryption to hide some area of the picture and ABE for encrypting that secret key has one problem when it is applied to a OSN: once a user has access to decrypt that piece of information, she might share the secret key and thus no more security will be provided. So, unlike [4] and [64] we do not rely on symmetric encryption together with ABE.

Our proposal, in comparison to [99], contemplates both DOSNs and OSNs. We do not need to include two more parties in the architecture such as a key server and a *Certified Authority (CA).* We do not need to create a dedicated application on the client's side to view the encrypted photo. We support both, JPEG, PNG and TIFF files. Additionally, we have tested our proposed solution based on different attributes on both the universe $\mathcal{U}$ and in the access policy $\mathcal{T}$.

Furthermore, it is worth mentioning that classical ABE approaches are based on a centralised assumption where a TP is in charge of distributing the keys of the schema and sets up the system. However this is infeasible in DOSNs because there were no practical solution if there are n-different authorities running the same cryptographic scheme and users from different authorities want to share out private information. Nonetheless, Rouselakis *et al.* proposed in [79] a decentralised and MA-ABE where different authorities spread all over the world can share information in a secure way by using an ABE scheme.

Another still open issue in MA-ABE is how to revocate attributes, i.e., how to generate again the users' secret keys once an attribute is not hold by a user anymore. In the literature there are some approaches such as using an expiration time in the access policy $\mathcal{T}$ or using specific cryptographic primitives [76]. However, in our approach we have solved it by running the *KeyGen* algorithm each time a photo is requested by a user.

VIII.6 Conclusions

In this paper we have proposed a solution for re-sharing photos securely on distributed social networks. We have used ABE to encrypt and decrypt the content of the picture that belongs to that person and thus, users can define different access control according to some policies previously defined over the same image. Moreover, as far as we know, this is the first solution that can be deployed into both decentralised and centralised social networks and we also allow different photograph's formats such as PNG, JPEG or TIFF. Finally, we have tested our solution on the distributed social network Diaspora, with one pod (centralised mode) and more than three pods (decentralised mode), a hundred of attributes each and the evaluations show that our solution can encrypt/decrypt images in less than two seconds.

ABE guarantees, by construction, that only those users having the "right" attributes can decrypt a ciphertext previously encrypted with a certain access policy aimed at users with those attributes. On the other hand, ABE does not ensure that users indeed have the attributes they claim to have. In most ABE proposed schemes researchers assume that there is a trusted party in charge of verifying that a user holds the attributes she claims to have. Though we do not explicit depend on this assumption, our proof-of-concept implementation in Diaspora comes with strong guarantees in this sense: the attributes of our policies are relationships between users and cannot be faked.[6] That said, our approach is more general and our policy description would in principle allow to define other attributes besides relationships in the OSN, like profession or location, which might be fields on a user profile and thus under control of the user. In this case we would need a trusted party to certify that the user has the attributes she claims to have.

Future Work. Currently there are no well-defined rules about who can encrypt which parts of a photo. In this work we impose the rule that no-one can re-encrypt areas of a picture that are already encrypted. This simple rule might not be enough from the point of view of usability. It might still lead to undesirable behaviours. For instance, imagine that Alice uploads a photo of herself without encryption. Later Bob—who has access to the photo—decides to encrypt some part of it so that only he can see the photo. In other words, now Alice cannot see parts of the photo that she uploaded.

[6]We do assume that Diaspora is correctly implemented, and that users cannot access and modify the corresponding database.

This authorisation problems go beyond the scope of this paper and require a detailed analysis of the interactions that can be performed in the social network together with the encryption algorithms. There are formal techniques to attack this problem, in particular to encode privacy settings of social networks and formally reason about them [31, 69, 71]. We plan to formalise our solution in order to precisely define which actions are allowed and prove that no undesirable behaviours can occur.

Bibliography

[1] Alessandro Acquisti and Ralph Gross. 'Imagined Communities: Awareness, Information Sharing, and Privacy on the Facebook'. In: *Proceedings of the 6th International Workshop Privacy Enhancing Technologies, PET'06*. Vol. 4258. LNCS. 2006, pp. 36–58. ISBN: 3-540-68790-4. DOI: 10.1007/11957454_3.

[2] *Alexa-Ranking.* http://www.alexa.com/topsites [Accessed: 2017-09-20].

[3] Guillaume Aucher, Guido Boella, and Leendert Torre. 'A dynamic logic for privacy compliance'. In: *Journal of Artificial Intelligence and Law* 19.2-3 (2011), pp. 187–231. ISSN: 0924-8463. DOI: 10.1007/s10506-011-9114-3.

[4] Randolph Baden, Adam Bender, Neil Spring, Bobby Bhattacharjee, and Daniel Starin. 'Persona: an online social network with user-defined privacy'. In: *Proceedings of the 9th ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, CCR'09*. 2009, pp. 135–146. DOI: 10.1145/1592568.1592585.

[5] Musard Balliu. 'A Logic for Information Flow Analysis of Distributed Programs'. In: *Proceedings of 18th Nordic Conference Secure IT Systems, NordSec'13*. Vol. 8208. LNCS. 2013, pp. 84–99. ISBN: 978-3-642-41487-9. DOI: 10.1007/978-3-642-41488-6_6.

[6] Lujo Bauer, Lorrie Faith Cranor, Saranga Komanduri, Michelle L. Mazurek, Michael K. Reiter, Manya Sleeper, and Blase Ur. 'The post anachronism: the temporal dimension of facebook privacy'. In: *Proceedings of the 12th annual ACM Workshop on Privacy in the Electronic Society, WPES'13*. 2013, pp. 1–12. ISBN: 978-1-4503-2485-4. DOI: 10.1145/2517840.2517859.

[7] Johan van Benthem, Jan van Eijck, and Barteld Kooi. 'Logics of communication and change'. In: *Journal of Information and computation* 204.11 (2006), pp. 1620–1662. DOI: 10.1016/j.ic.2006.04.006.

[8] Ido Ben-Zvi and Yoram Moses. 'Agent-time epistemics and coordination'. In: *Proceedings of the 5th Indian Conference on Logic and Its Applications, ICLA'13*. Vol. 7750. LNCS. 2013, pp. 97–108. DOI: `10.1007/978-3-642-36039-8_9`.

[9] Bostom.com. *Harvard student loses Facebook internship after pointing out privacy flaws.* http://www.boston.com/news/nation/2015/08/12/harvard-student-loses-facebook-internship-after-pointing-out-privacy-flaws/. [Accessed: 2017-09-20]. 2015.

[10] Danah Boyd and Nicole B. Ellison. 'Social network sites: Definition, history, and scholarship'. In: *Journal of Computer-Mediated Communication* 13.1 (2007), pp. 210–230. DOI: `10.1111/j.1083-6101.2007.00393.x`.

[11] Nathan Bronson, Zach Amsden, George Cabrera, Prasad Chakka, Peter Dimov, Hui Ding, Jack Ferris, Anthony Giardullo, Sachin Kulkarni, Harry Li, Mark Marchukov, Dmitri Petrov, Lovro Puzar, Yee Jiun Song, and Venkat Venkataramani. 'TAO: Facebook's Distributed Data Store for the Social Graph'. In: *Proceedings of the 2013 USENIX Annual Technical Conference, USENIX ATC'13*. 2013, pp. 49–60. ISBN: 978-1-931971-01-0.

[12] Glenn Bruns, Philip WL Fong, Ida Siahaan, and Michael Huth. 'Relationship-based access control: its expression and enforcement through hybrid logic'. In: *Proceedings of the 2nd ACM conference on Data and Application Security and Privacy, CODASPY'12*. 2012, pp. 117–124. ISBN: 978-1-4503-1091-8. DOI: `10.1145/2133601.2133616`.

[13] Sonja Buchegger, Doris Schiöberg, Le-Hung Vu, and Anwitaman Datta. 'PeerSoN: P2P Social Networking: Early Experiences and Insights'. In: *Proceedings of the 2nd ACM EuroSys Workshop on Social Network Systems, SNS'09*. 2009, pp. 46–52. ISBN: 978-1-60558-463-8. DOI: `10.1145/1578002.1578010`.

[14] Twitter Help Center. *Protecting and unprotecting your Tweets.* https://support.twitter.com/articles/20169886. [Accessed: 2017-09-20]. 2016.

[15] Zoé Christoff and Jens Ulrik Hansen. 'A logic for diffusion in social networks'. In: *Journal of Applied Logic* 13.1 (2015), pp. 48–77. ISSN: 1570-8683. DOI: `10.1016/j.jal.2014.11.011`.

[16] Zoé Christoff and Jens Ulrik Hansen. 'Dynamic social networks logic'. In: *Journal of Applied Logic (to appear), available as ILLC Prepublication Series Report PP-2014-09* (2014).

[17] Christian Colombo, Gordon J. Pace, and Gerardo Schneider. 'Dynamic Event-Based Runtime Monitoring of Real-Time and Contextual Properties'. In: *Proceedings of 13th International Workshop on Formal Methods for Industrial Critical Systems, FMICS'08*. Vol. 5596. LNCS. 2009, pp. 135–149. ISBN: 978-3-642-03239-4. DOI: 10.1007/978-3-642-03240-0_13.

[18] Christian Colombo, Gordon J. Pace, and Gerardo Schneider. 'LARVA — Safer Monitoring of Real-Time Java Programs (Tool Paper)'. In: *Proceedings of the 7th IEEE International Conference on Software Engineering and Formal Methods, SEFM'09*. 2009, pp. 33–37. ISBN: 978-0-7695-3870-9. DOI: 10.1109/SEFM.2009.13.

[19] Leucio Antonio Cutillo, Refik Molva, and Thorsten Strufe. 'Safebook: A privacy-preserving online social network leveraging on real-life trust'. In: *IEEE Communications Magazine* 47.12 (2009), pp. 94–101. DOI: 10.1109/MCOM.2009.5350374.

[20] Anupam Datta, Jeremiah Blocki, Nicolas Christin, Henry DeYoung, Deepak Garg, Limin Jia, Dilsun Kirli Kaynar, and Arunesh Sinha. 'Understanding and Protecting Privacy: Formal Semantics and Principled Audit Mechanisms'. In: *Proceedings of the 7th International Conference on Information Systems Security, ICISS'11*. Vol. 7093. LNCS. 2011, pp. 1–27. ISBN: 978-3-642-25559-5. DOI: 10.1007/978-3-642-25560-1_1.

[21] *$\mathcal{PPF}$ Diaspora**. Test pod: https://ppf-diaspora.raulpardo.org.
Code: https://github.com/raulpardo/ppf-diaspora [Accessed: 2017-09-20].

[22] *Diaspora**. https://diasporafoundation.org/ [Accessed: 2017-09-20].

[23] Daniel J. Dougherty, Kathi Fisler, and Shriram Krishnamurthi. 'Specifying and Reasoning About Dynamic Access-Control Policies'. In: *Proceedings of the 3rd International Joint Conference on Automated Reasoning, IJCAR'06*. Vol. 4130. LNCS. 2006, pp. 632–646. ISBN: 3-540-37187-7. DOI: 10.1007/11814771_51.

[24] Azar Eftekhar, Chris Fullwood, and Neil Morris. 'Capturing personality from Facebook photos and photo-related activities: How much exposure do you need?' In: *Journal of Computers in Human Behavior* 37 (2014), pp. 162–170. ISSN: 0747-5632. DOI: 10.1016/j.chb.2014.04.048.

[25] Jan van Eijck. 'DEMO – A Demo of Epistemic Modelling'. In: *Proceedings of the 7th Augustus de Morgan Workshop, AMW'07*. 2007, pp. 305–363.

[26] Nicole B. Ellison, Jessica Vitak, Charles Steinfield, Rebecca Gray, and Cliff Lampe. 'Negotiating Privacy Concerns and Social Capital Needs in a Social Media Environment'. In: *Privacy Online - Perspectives on Privacy and Self-Disclosure in the Social Web.* 2011, pp. 19–32. DOI: 10.1007/978-3-642-21521-6_3.

[27] Kayhan Erciyes. *Complex Networks: An Algorithmic Perspective.* CRC Press, Inc., 2014. ISBN: 978-1466571662.

[28] *Facebook.* https://www.facebook.com/ [Accessed: 2017-09-20].

[29] Ronald Fagin, Joseph Y Halpern, Yoram Moses, and Moshe Y Vardi. *Reasoning about knowledge.* Vol. 4. MIT press Cambridge, 2003. ISBN: 978-0262562003.

[30] *FlockDB: A distributed, fault-tolerant graph database.* https://github.com/twitter/flockdb [Accessed: 2017-09-20].

[31] Philip W.L. Fong. 'Relationship-based Access Control: Protection Model and Policy Language'. In: *Proceedings of the 1st ACM Conference on Data and Application Security and Privacy, CODASPY'11.* 2011, pp. 191–202. ISBN: 978-1-4503-0466-5. DOI: 10.1145/1943513.1943539.

[32] PhilipW.L. Fong, Mohd Anwar, and Zhen Zhao. 'A Privacy Preservation Model for Facebook-Style Social Network Systems'. In: *Proceedings of the 14th European Symposium on Research in Computer Security, ESORICS'09.* Vol. 5789. LNCS. Springer, 2009, pp. 303–320. ISBN: 978-3-642-04443-4. DOI: 10.1007/978-3-642-04444-1_19.

[33] Peter Gammie and Ron van der Meyden. 'MCK: Model Checking the Logic of Knowledge'. In: *Proceedings of the 16th International Conference on Computer Aided Verification, CAV'04.* Vol. 3114. LNCS. 2004, pp. 479–483. ISBN: 978-3-540-22342-9. DOI: 10.1007/978-3-540-27813-9_41.

[34] G. Scott Graham and Peter J. Denning. 'Protection: principles and practice'. In: *American Federation of Information Processing Societies: AFIPS Conference Proceedings: 1972 Spring Joint Computer Conference.* 1972, pp. 417–429. DOI: 10.1145/1478873.1478928.

[35] Benjamin Greschbach, Gunnar Kreitz, and Sonja Buchegger. 'The devil is in the metadata - New privacy challenges in Decentralised Online Social Networks'. In: *Proceedings of the 10th Annual IEEE International Conference on Pervasive Computing and Communications, PerCom'12.* 2012, pp. 333–339. ISBN: 978-1-4673-0905-9. DOI: 10.1109/PerComW.2012.6197506.

[36] Ralph Gross and Alessandro Acquisti. ‘Information revelation and privacy in online social networks’. In: *Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society, WPES’05.* 2005, pp. 71–80. ISBN: 1-59593-228-3. DOI: `10.1145/1102199.1102214`.

[37] G. L. Guernic. ‘Automaton-based Confidentiality Monitoring of Concurrent Programs’. In: *Proceedings of 20th IEEE Computer Security Foundations Symposium, CSF’07.* 2007, pp. 218–232. ISBN: 0-7695-2819-8. DOI: `10.1109/CSF.2007.10`.

[38] Joseph Y Halpern and Kevin R O’Neill. ‘Secrecy in multiagent systems’. In: *ACM Transactions on Information and System Security* 12.1 (2008), p. 5. DOI: `10.1145/1410234.1410239`.

[39] Joseph Y. Halpern, Dov Samet, and Ella Segev. ‘Defining Knowledge in Terms of Belief: The Modal Logic Perspective’. In: *Journal of the Review of Symbolic Logic* 2.3 (2009), pp. 469–487. DOI: `10.1017/S1755020309990141`.

[40] Andrew K. Hirsch and Michael R. Clarkson. ‘Belief Semantics of Authorization Logic’. In: *Proceedings of the 20th ACM SIGSAC Conference on Computer & Communications Security, CCS’13.* 2013, pp. 561–572. ISBN: 978-1-4503-2477-9. DOI: `10.1145/2508859.2516667`.

[41] Maritza Johnson, Serge Egelman, and Steven M. Bellovin. ‘Facebook and Privacy: It’s Complicated’. In: *Proceedings of the 8th Symposium on Usable Privacy and Security, SOUPS’12.* 2012, pp. 1–15. ISBN: 978-1-4503-1532-6. DOI: `10.1145/2335356.2335369`.

[42] Aliaksandr Lazouski, Fabio Martinelli, and Paolo Mori. ‘Usage control in computer security: A survey’. In: *Journal of Computer Science Review* 4.2 (2010), pp. 81–99. DOI: `10.1016/j.cosrev.2010.02.002`.

[43] Gurvan Le Guernic, Anindya Banerjee, Thomas Jensen, and David A. Schmidt. ‘Automata-Based Confidentiality Monitoring’. In: *Proceedings of the 11th Asian Computing Science Conference on Secure Software and Related Issues, ASIAN’06.* 2007, pp. 75–89. ISBN: 978-3-540-77505-8. DOI: `10.1007/978-3-540-77505-8_7`.

[44] Daniel Le Métayer. ‘Privacy by Design: A Formal Framework for the Analysis of Architectural Choices’. In: *Proceedings of the 3rd ACM Conference on Data and Application Security and Privacy, CODASPY’13.* 2013, pp. 95–104. ISBN: 978-1-4503-1890-7. DOI: `http://doi.acm.org/10.1145/2435349.2435361`.

[45] Amanda Lenhart, Kristen Purcell, Aaron Smith, and Kathryn Zickuhr. ‘Social Media & Mobile Internet Use among Teens and Young Adults. Millennials.’ In: *Pew Internet & American Life Project* (2010).

[46] Allison Lewko and Brent Waters. 'Decentralizing Attribute-based Encryption'. In: *Proceedings of the 30th Annual International Conference on Theory and Applications of Cryptographic Techniques: Advances in Cryptology, EUROCRYPT'11*. Vol. 6632. LNCS. 2011, pp. 568–588. ISBN: 978-3-642-20464-7.

[47] Ninghui Li and Mahesh V. Tripunitara. 'On Safety in Discretionary Access Control'. In: *Proceedings of the 26th IEEE Symposium on Security and Privacy, S&P'05*. 2005, pp. 96–109. ISBN: 0-7695-2339-0. DOI: `10.1109/SP.2005.14`.

[48] Kaitai Liang, Joseph K. Liu, Rongxing Lu, and Duncan S. Wong. 'Privacy Concerns for Photo Sharing in Online Social Networks'. In: *Journal of IEEE Internet Computing* 19.2 (2015), pp. 58–63. ISSN: 1089-7801. DOI: `10.1109/MIC.2014.107`.

[49] Jay Ligatti, Lujo Bauer, and David Walker. 'Edit automata: Enforcement mechanisms for run-time security policies'. In: *Journal of Information Security* 4 (2005), pp. 2–16.

[50] Eden Litt and Eszter Hargittai. 'Smile, snap, and share? A nuanced approach to privacy and online photo-sharing'. In: *Poetics* 42 (2014), pp. 1–21. ISSN: 0304-422X. DOI: `10.1016/j.poetic.2013.10.002`.

[51] Yabing Liu, Krishna P. Gummadi, Balachander Krishnamurthy, and Alan Mislove. 'Analyzing Facebook Privacy Settings: User Expectations vs. Reality'. In: *Proceedings of the 11th ACM SIGCOMM Conference on Internet Measurement Conference, IMC'11*. 2011, pp. 61–70. ISBN: 978-1-4503-1013-0. DOI: `10.1145/2068816.2068823`.

[52] Katharina Lobinger. 'Photographs as things – photographs of things. A texto-material perspective on photo-sharing practices'. In: *Journal of Information, Communication & Society* 19.4 (2016), pp. 475–488.

[53] Alessio Lomuscio, Hongyang Qu, and Franco Raimondi. 'MCMAS: A model checker for the verification of multi-agent systems'. In: *Proceedings of the 21st International Conference on Computer Aided Verification, CAV'09*. Vol. 5643. LNCS. 2009, pp. 682–688. ISBN: 978-3-642-02657-7. DOI: `10.1007/978-3-642-02658-4_55`.

[54] Alessio Lomuscio, Hongyang Qu, and Franco Raimondi. 'MCMAS: an open-source model checker for the verification of multi-agent systems'. In: *International Journal on Software Tools for Technology Transfer* 19.1 (2017), pp. 9–30. DOI: `10.1007/s10009-015-0378-x`.

[55] Alessio Lomuscio and Mark Ryan. 'On the Relation between Interpreted Systems and Kripke Models'. In: *Agents and Multi-Agent Systems Formalisms, Methodologies, and Applications.* Vol. 1441. LNCS. 1997, pp. 46–59. ISBN: 3-540-64769-4.

[56] M. Madejski, M. Johnson, and S.M. Bellovin. 'A study of privacy settings errors in an online social network'. In: *Proceedings of the 10th IEEE International Conference on Pervasive Computing and Communication Workshops, PerCom'12.* 2012, pp. 340–345. ISBN: 978-1-4673-0905-9. DOI: `10.1109/PerComW.2012.6197507`.

[57] Michelle Madejski, Maritza Lupe Johnson, and Steven Michael Bellovin. 'The failure of online social network privacy settings'. In: Columbia University Computer Science Technical Reports (2011). Technical report: CUCS-010-11. DOI: `10.7916/D8NG4ZJ1`.

[58] Aqdas Malik, Amandeep Dhir, and Marko Nieminen. 'Uses and Gratifications of digital photo sharing on Facebook'. In: *Journal of Telematics and Informatics* 33.1 (2016), pp. 129–138. ISSN: 0736-5853. DOI: `10.1016/j.tele.2015.06.009`.

[59] John-Jules Ch Meyer and Wiebe Van Der Hoek. *Epistemic Logic for AI and Computer Science.* Cambridge University Press, 1995. ISBN: 052146014X.

[60] Mainack Mondal, Peter Druschel, Krishna P. Gummadi, and Alan Mislove. 'Beyond Access Control: Managing Online Privacy via Exposure'. In: *Proceedings of NDSS Workshop on Usable Security Workshop, USEC'14.* 2014. ISBN: 1-891562-37-1. DOI: `10.14722/usec.2014.23046`.

[61] *Neo4j decreases development time-to-market for LinkedIn's Chitu App.* https://neo4j.com/case-studies/linkedin-china/ [Accessed: 2017-09-20]. 2017.

[62] Facebook's newsroom. *Improving the Experience When Relationships End.* https://newsroom.fb.com/news/2015/11/improving-the-experience-when-relationships-end/. [Accessed: 2017-09-20]. 2015.

[63] Facebook's newsroom. *Making It Easier to Share With Who You Want.* https://newsroom.fb.com/news/2014/05/making-it-easier-to-share-with-who-you-want/. [Accessed: 2017-09-20]. 2014.

[64] Shirin Nilizadeh, Sonia Jahid, Prateek Mittal, Nikita Borisov, and Apu Kapadia. 'Cachet: A Decentralized Architecture for Privacy Preserving Social Networking with Caching'. In: *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies, CoNEXT'12.* 2012, pp. 337–348. ISBN: 978-1-4503-1775-7. DOI: `10.1145/2413176.2413215`.

[65] Rajesh Nishtala, Hans Fugal, Steven Grimm, Marc Kwiatkowski, Herman Lee, Harry C. Li, Ryan McElroy, Mike Paleczny, Daniel Peek, Paul Saab, David Stafford, Tony Tung, and Venkateshwaran Venkataramani. 'Scaling Memcache at Facebook'. In: *Proceedings of the 10th USENIX Symposium on Networked Systems Design and Implementation NSDI'13*. 2013, pp. 385–398. ISBN: 978-1-931971-00-3.

[66] Raúl Pardo. *Formalising Privacy Policies for Social Networks*. Pages 102. Licentiate thesis. Department of Computer Science and Engineering, Chalmers University of Technology, 2015.

[67] Raúl Pardo, Musard Balliu, and Gerardo Schneider. 'Formalising privacy policies in social networks'. In: *Journal of Logical and Algebraic Methods in Programming* (2017). ISSN: 2352-2208. DOI: `10.1016/j.jlamp.2017.02.008`.

[68] Raúl Pardo, Christian Colombo, Gordon J. Pace, and Gerardo Schneider. 'An Automata-Based Approach to Evolving Privacy Policies for Social Networks'. In: *Proceedings of the 16th International Conference on Runtime Verification, RV'16*. Vol. 10012. LNCS. 2016, pp. 285–301. ISBN: 978-3-319-46981-2. DOI: `10.1007/978-3-319-46982-9_18`.

[69] Raúl Pardo, Ivana Kellyérová, César Sánchez, and Gerardo Schneider. 'Specification of Evolving Privacy Policies for Online Social Networks'. In: *Proceedings of the 23rd International Symposium on Temporal Representation and Reasoning, TIME'16*. IEEE, 2016, pp. 70–79. ISBN: 978-1-5090-3825-1. DOI: `10.1109/TIME.2016.15`.

[70] Raúl Pardo, César Sánchez, and Gerardo Schneider. 'Timed Epistemic Knowledge Bases for Social Networks (Extended Version)'. In: *ArXiv e-prints* (2017). eprint: `1708.04070` (cs.LO).

[71] Raúl Pardo and Gerardo Schneider. 'A Formal Privacy Policy Framework for Social Networks'. In: *Proceedings of the 12th International Conference on Software Engineering and Formal Methods, SEFM'14*. Vol. 8702. LNCS. 2014, pp. 378–392. ISBN: 978-3-319-10430-0. DOI: `10.1007/978-3-319-10431-7_30`.

[72] Raúl Pardo and Gerardo Schneider. 'Model Checking Social Network Models'. In: *Proceedings of the Eighth International Symposium on Games, Automata, Logics and Formal Verification, GandALF'17*. Vol. 256. EPTCS. 2017, pp. 238–252. DOI: `10.4204/EPTCS.256.17`.

[73] Pablo Picazo-Sanchez, Raúl Pardo, and Gerardo Schneider. ‘Secure Photo Sharing in Social Networks’. In: *Proceedings of the 32nd International Conference on ICT Systems Security and Privacy Protection, IFIP SEC 2017*. Vol. 502. 2017, pp. 79–92. ISBN: 978-3-319-58469-0. DOI: `10.1007/978-3-319-58469-0_6`.

[74] Jan Plaza. ‘Logics of public communications’. In: *Synthese* 158.2 (2007), pp. 165–179. ISSN: 0039-7857. DOI: `10.1007/s11229-007-9168-7`.

[75] Riccardo Pucella. ‘Knowledge and Security’. In: *ArXiv e-prints* (2013). eprint: `1305.0876` (cs.CR).

[76] Huiling Qian, Jiguo Li, Yichen Zhang, and Jinguang Han. ‘Privacy-preserving personal health record using multi-authority attribute-based encryption with revocation’. In: *International Journal of Information Security* 14.6 (2015), pp. 487–497. ISSN: 1615-5270. DOI: `10.1007/s10207-014-0270-9`.

[77] Moo-Ryong Ra, Ramesh Govindan, and Antonio Ortega. ‘P3: Toward Privacy-Preserving Photo Sharing’. In: *Proceedings of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI’13)*. 2013, pp. 515–528. ISBN: 978-1-931971-00-3.

[78] Moritz Riesner, Michael Netter, and Günther Pernul. ‘An Analysis of Implemented and Desirable Settings for Identity Management on Social Networking Sites’. In: *Proceedings of the 7th International Conference on Availability, Reliability and Security, ARES’12*. 2012, pp. 103–112. ISBN: 978-1-4673-2244-7. DOI: `10.1109/ARES.2012.20`.

[79] Yannis Rouselakis and Brent Waters. ‘Efficient Statically-Secure Large-Universe Multi-Authority Attribute-Based Encryption’. In: *Proceedings of the 19th International Conference on Financial Cryptography and Data Security, FC’15*. Vol. 8975. LNCS. 2015, pp. 315–332. DOI: `10.1007/978-3-662-47854-7_19`.

[80] Ji Ruan and Michael Thielscher. ‘A logic for knowledge flow in social networks’. In: *Proceedings of the 24th Australasian Joint Conference on Advances in Artificial Intelligence, AI’12*. Vol. 7106. LNCS. 2011, pp. 511–520. ISBN: 978-3-642-25831-2. DOI: `10.1007/978-3-642-25832-9_52`.

[81] Xin Ruan, Chuan Yue, and Haining Wang. ‘Unveiling Privacy Setting Breaches in Online Social Networks’. In: *Proceedings of the 9th International Conference on Security and Privacy in Communication Networks, SecureComm’13*. Vol. 127. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. 2013, pp. 323–341. DOI: `10.1007/978-3-319-04283-1_20`.

[82] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. 'Role-Based Access Control Models'. In: *Journal of IEEE Computer* 29.2 (1996), pp. 38–47. DOI: `10.1109/2.485845`.

[83] Fred B. Schneider. 'Enforceable Security Policies'. In: *ACM Transactions on Information and System Security* 3.1 (2000), pp. 30–50. ISSN: 1094-9224.

[84] Jeremy Seligman, Fenrong Liu, and Patrick Girard. 'Facebook and the Epistemic Logic of Friendship'. In: *Proceedings of the 14th Conference on Theoretical Aspects of Rationality and Knowledge, TARK'13.* 2013. ISBN: 978-0-615-74716-3.

[85] Jeremy Seligman, Fenrong Liu, and Patrick Girard. 'Knowledge, friendship and social announcement'. In: *Proceedings of the Tsinghua Logic Conference.* 2013.

[86] *Snaptchat.* https://www.snapchat.com/ [Accessed: 2017-09-20].

[87] Richard Snodgrass and Ilsoo Ahn. 'Temporal Databases'. In: *Journal of IEEE Computer* 19.9 (1986), pp. 35–42. ISSN: 0018-9162. DOI: `10.1109/MC.1986.1663327`.

[88] Statista. *Facebook statistics.* https://www.statista.com/statistics/264810/number-of-monthly-active-facebook-users-worldwide/. [Accessed: 2017-09-20]. 2016.

[89] Fred Stutzman and Jacob Kramer-Duffield. 'Friends Only: Examining a Privacy-enhancing Behavior in Facebook'. In: *Proceedings of the 28th SIGCHI Conference on Human Factors in Computing Systems, CHI'10.* 2010, pp. 1553–1562. ISBN: 978-1-60558-929-9. DOI: `10.1145/1753326.1753559`.

[90] Sanaz Taheri-Boshrooyeh, Alptekin Küpçü, and Öznur Özkasap. 'Security and Privacy of Distributed Online Social Networks'. In: *Proceedings of the 35th IEEE International Conference on Distributed Computing Systems Workshops, ICDCS'15.* 2015, pp. 112–119. ISBN: 978-1-4673-7303-6. DOI: `10.1109/ICDCSW.2015.30`.

[91] The Guardian. *As fake news takes over Facebook feeds, many are taking satire as fact.* https://www.theguardian.com/media/2016/nov/17/facebook-fake-news-satire. [Accessed: 2017-09-20]. 2016.

[92] The Guardian. *How to solve Facebook's fake news problem: experts pitch their ideas.* https://www.theguardian.com/technology/2016/nov/29/facebook-fake-news-problem-experts-pitch-ideas-algorithms. [Accessed: 2017-09-20]. 2016.

[93] The Guardian. *Obama is worried about fake news on social media–and we should be too.* https://www.theguardian.com/media/2016/nov/20/barack-obama-facebook-fake-news-problem. [Accessed: 2017-09-20]. 2016.

[94] *Twitter, Inc.* https://twitter.com/ [Accessed: 2017-09-20].

[95] Brent Waters. 'Ciphertext-policy Attribute-based Encryption: An Expressive, Efficient, and Provably Secure Realization'. In: *Proceedings of the 14th International Conference on Practice and Theory in Public Key Cryptography Conference on Public Key Cryptography, PKC'11*. Vol. 6571. LNCS. 2011, pp. 53–70. ISBN: 978-3-642-19378-1. DOI: 10.1007/978-3-642-19379-8_4.

[96] Daniel J. Weitzner, Harold Abelson, Tim Berners-Lee, Joan Feigenbaum, James A. Hendler, and Gerald J. Sussman. 'Information accountability'. In: *Communincations of the ACM* 51.6 (2008), pp. 82–87. DOI: 10.1145/1349026.1349043.

[97] Bożena Woźna and Alessio Lomuscio. 'A Logic for Knowledge, Correctness, and Real Time'. In: *Proceedings of the 5th International Workshop on Computational Logic in Multi-Agent Systems, CLIMA'04*. Vol. 3487. LNCS. 2004, pp. 1–15. DOI: 10.1007/11533092_1.

[98] Zuojun Xiong, Thomas Ågotnes, Jeremy Seligman, and Rui Zhu. 'Towards a Logic of Tweeting'. In: *Proceedings of the 6th International Workshop on Logic, Rationality, and Interaction, LORI'17*. Vol. 10455. LNCS. 2017, pp. 49–64. DOI: 10.1007/978-3-662-55665-8_4.

[99] Lin Yuan, David Mc Nally, Alptekin Küpçü, and Touradj Ebrahimi. 'Privacy-Preserving Photo Sharing based on a Public Key Infrastructure'. In: *SPIE Optical Engineering + Applications*. Vol. 9599. Applications of Digital Image Processing XXXVIII. 2015. DOI: 10.1117/12.2190458.

[100] Marc van Zee, Dragan Doder, Mehdi Dastani, and Leendert W. N. van der Torre. 'AGM Revision of Beliefs about Action and Time'. In: *Proceedings of the 24th International Joint Conference on Artificial Intelligence, IJCAI'15*. 2015, pp. 3250–3256. ISBN: 978-1-57735-738-4.

[101] Lan Zhang, Taeho Jung, Cihang Liu, Xuan Ding, Xiang-Yang Li, and Yunhao Liu. 'POP: Privacy-Preserving Outsourced Photo Sharing and Searching for Mobile Devices'. In: *Proceedings of the 35th IEEE International Conference on Distributed Computing Systems, ICDCS'15*. 2015, pp. 308–317. DOI: 10.1109/ICDCS.2015.39.