THESIS FOR THE DEGREE OF LICENTIATE OF ENGINEERING

Formalising Privacy Policies for Social Networks

RAÚL PARDO

CHALMERS | GÖTEBORG UNIVERSITY

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY AND GÖTEBORG UNIVERSITY

Göteborg, Sweden 2015

Formalising Privacy Policies for Social Networks
RAÚL PARDO

Technical Report 135L
ISSN 1652-876X
Department of Computer Science and Engineering
Research group: Formal Methods & Language-based Security

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY and GÖTEBORG UNIVERSITY
SE-412 96 Göteborg
Sweden
Telephone +46 (0)31-772 1000

Printed at Chalmers
Göteborg, Sweden 2015

Abstract

Social Network Services (SNSs) have changed the way people communicate, bringing many benefits but also the possibility of new threats. Privacy is one of them. We present here a framework to write privacy policies for SNSs and to reason about such policies in the presence of events making the network to evolve. The framework includes a model of SNSs, a logic to specify properties and reasoning 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 to infer knowledge from previously acquired one. 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 any possible event of the system). We also show how Twitter and Facebook are not privacy-preserving in the presence of additional natural privacy policies.

Acknowledgments

The work in this thesis would not have been possible without the help of many people.

First of all, I would like to thank my advisor Gerardo Schneider for his guidance and insightful discussions, which normally lead to new and more challenging research questions. I am surprised how much I have learnt in such a short period of time, and this is mainly because of you. I am looking forward to continuing working with you during the upcoming years towards my PhD. Most importantly, I am very grateful for our friendship, it is great when your advisor is also a good friend.

Thanks to Musard Balliu, who has had a strong influence in this work and for his comments on earlier drafts of this thesis. I am also grateful to Andrei Sabelfeld for his insights about my research and for the comments on a previous version of this thesis. I would also like to thank the all members of the Formal Methods and ProSec groups, especially to Dave Sands and Wolfgang Ahrendt, for sharing with me new viewpoints and interesting ideas after presenting my research in the groups' semaniars.

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.

I am very grateful to all the new friends I have made in Sweden. Thanks to Pablo, Bart, Evgeny, Mauricio, Danielito, Daniel, Hamid, Luciano, Anders, Steven, Elena, John, Simon, Willard, Michał, Nick,. . .[1] for being there to hang out, go climbing, go dancing, barbecues, afterworks, etcetera. Thanks you guys also for all the experiences you share with me, which are not only fun but also making me to grow as a person.

Quiero agradecer a toda mi familia su apoyo incondicional, especialmente a mis padres Diego y Paqui, ya que gracias a su esfuerzo he podido llegar hasta aquí. Muchas gracias también a mis hermanas Carmen y Llanos, que siempre están ahí y hacen que la distancia casi no se note.

Por supuesto no me olvido de amigos Rafa, Callejas, Juanma, Kike, Lucio, Andrés, Isra, Villa, Bea, Pedro, Laura, 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.

[1]I apoligise beforehand to those I forgot to mention.

Contents

Chapter 1

Introduction

1.1 Do We Need Privacy?

The concept of privacy has historical origins already in Ancient Greece. The philosopher Aristotle described the distinction between the public sphere and political activity, the *polis*, and the privacy or domestic sphere of the family, the *oikos*. How privacy should be valued and preserved has extensively been discussed in various cultures. Unfortunately, privacy is not a static notion, since it varies over time making its definition a controversial and cumbersome issue. Even today we are looking for an appropriate definition of privacy. Warren & Brandeis provided one of the first "formal" definitions of privacy as *the right to be alone* [34]. However, the complex and ambiguous nature of privacy has turned it into an umbrella term for a variety of loosely related issues and problems [33]. Despite of not having a precise notion, privacy is understood as a necessary condition for individual autonomy, identity, and integrity [8, 35].

Furthermore, privacy is accepted as a basic human need. It is also stated as the 12th article of *The Universal Declaration of Human Rights* [6], which is a list of moral principles and norms inherent in all human beings. Privacy is closely linked to more basic human values such as autonomy, freedom or self-fulfilment of an individual. Think for a minute whether your behaviour would be altered if you knew that you are being recorded 24/7. Controlling people's privacy is a synonym of power, e.g. companies often warn their employees that their network traffic will be constantly monitored with the objective of avoiding their going to leisure websites during working hours. Later the network traffic can be monitor or not, but the behaviour of most of the employees will be conditioned by the fact that they think that they are being monitored.

In particular, privacy is related to the right of *self-determination* defined as “the right to freely determine what is necessary and desirable for a fulfilling and meaningful life and to freely pursue one’s social, cultural, political, and economic development” [13]. Self-determination can also be expressed in terms of data protection. In this case, it is called *informational self-determination*. Similarly to self-determination, in the informational variant it is expressed that an individual should be in full control of her information. Informational self-determination is not only related to basic human needs, but also is explicitly expressed in Data Protection directive of the European Union [2].

Online services, such as social networks, have created lots of unforeseen breaches of privacy (we describe some of them in the following section), since many of them were not designed with privacy mechanisms in mind. Do you think that Mark Zuckerberg could have anticipated all the privacy issues that Facebook is dealing with today? Social networks handle an enormous amount of personal information, but the privacy protections that the users are offered with do not give people enough control over their data. We believe that people should be in full control over their private information. In this thesis, we focus on providing a framework for developers to cope with all the functionalities of a social network, while always having present privacy implications, so that people are offered with tools that will enhance the control they have over the information they share in social networks.

1.2 The Problem of Privacy in Social Networks

Over the past decade, the use of the social networks like Facebook and Twitter, just to mention two of the most popular ones, has increased at the point of becoming ubiquitous. Many people access *Social Networks Services* (SNSs) on a daily basis; e.g. to read the news, share pictures with their friends or check upcoming events. Nearly 70% of the Internet users are active on SNSs, as shown by a recent survey [21].

At the same time, privacy in SNSs has become a global concern. Influential newspapers all over the world, such as The Guardian, The New York Times or Wired, publish articles which show how the privacy of SNSs users is compromised. For instance, a few months ago The Guardian published an article related to a privacy flaw in the Facebook's application programming interface (API), which allows any developer to get personal information from Facebook users [3]. The problem lies on the privacy settings of the system. In Facebook there is a privacy setting called "Who can find me?", which determines who can find a user by her phone number. By default, it is set to "Everyone/Public", meaning that anyone can find another user by their mobile phone number. This default setting already has privacy implications, because just by registering a user mobile phone number that user's private information is accessible by any other user. If the user is not happy with this disclosure of information, then she has to change the privacy setting explicitly. This type of choice is called *opt-out* choice, meaning that users are *in* by default, but they have the possibility to opt-out. The main problem here is that normally users are not informed about which privacy settings they have activated by default, which leads to a violation of their privacy, since they are not aware with whom they are sharing their personal data. Despite of the repercussion it had on the media, Facebook developers answer to this problem by saying that they have monitoring tools that prevent an abusive used of their API.

Managing privacy settings is not an easy task. There exist many tutorials explaining how to handle privacy settings and describing the consequences of each choice, e.g. Facebook [4] or Instagram [7]. Why are privacy settings so complicated? For instance, in our previous example, is it reasonable to have a specific privacy setting which defines who can access your information using your mobile phone number? It requires a lot of knowledge from the user perspective. Most of the time, users are not aware of all the features and details of the SNS, therefore it is not very convenient for the privacy protection mechanism to rely on the users' knowledge about the behaviour of the SNS. Moreover, SNSs follow the so called *perpetual development*, which means that the system is never finished. Developers add new features almost on a daily basis, which makes a very difficult task to control all the consequences of modifying the SNS’ code.

1.2.1 Structure of SNSs

According to Boyd and Ellison [12] SNSs 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 SNS is the so called *social graph*, where users are represented by nodes and the edges are used to model the type of connection by which they are linked. Fig. 1.1 depicts an example of such a graph. In this social graph there are three users: Alice, Bob and Charlie. The straight line connecting Alice and Bob means that Alice and Bob are friends and the dotted line between Bob and Charlie shows that they are colleagues. Users are a common element in all SNSs, but connections may vary depending on the SNS. For instance, in Facebook the main connection between users is the *friendship* connection. In order for two Facebook users to become friends, it is required that one of them sends a friend request and the other accepts it. As a consequence, the friendship connection is symmetric, i.e. if Alice and Bob are friends in Facebook, then Alice is friend with Bob and vice versa. Not all connections are symmetric, a frequent connection between users is *follower*. It is present in several SNSs, e.g. Twitter, Instagram, Facebook and others. This connection is not required to be symmetric, meaning that Alice can follow Bob, but it is not necessary that Bob follows Alice.

1.2.2 Privacy Protection in SNSs

Normally, each SNSs offers a large variety of connections that users use to group other users, e.g. friends, family, co-workers, acquaintances, research, university and so forth. Users are closer

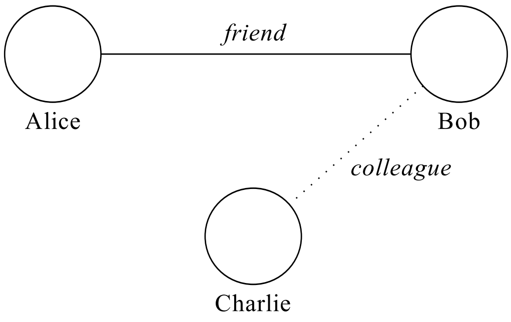

Figure 1.1: Social Graph Example

to people in some connections, like family, than to others, like acquaintances. Therefore, it makes sense for the privacy protection mechanism to depend on these connections. The current protection model for SNSs is based on the connections in the social graph. It allows users to decide which group of people has access to their private information. For example, a user can share the pictures of their child birthday with her family, and similarly, she can share the results of her last experiment with the users who belong to the research connection.

Intuitively, this protection model offers the control that users need for their personal information. However, it is susceptible to privacy breaches at different levels. Empirical studies have shown that the current privacy protections offered by SNSs are very far from the users' expectations [24, 20, 22, 25]. The main problems that users report regarding the current protection model in SNSs are related to privacy settings, which hereafter we will refer to as *privacy policies*. In particular, the main two problems that SNS users pointed out in the aforementioned studies are:

i) Privacy policies that SNSs offer are too coarse-grained.

ii) It is not easy to understand the result of activating a privacy policy.

Coarse-grained privacy policies

One of SNSs weaknesses is the inability for users to express desirable privacy policies. This is due to the lack of flexibility of the privacy policies SNSs provide to their users. Many desirable privacy policies are already offered by SNSs; for instance, in Facebook users can state polices like "Only my friends can see a post on my timeline" or "Whenever I am tagged, the picture should not be shown on my timeline unless I approve it". Many other policies, however, are not; although they might be important from a user's perspective. Again, using Facebook as an example, users cannot specify privacy policies like "I do not want be tagged in pictures by anyone other than myself" (P1) or "Nobody apart from myself can know my child's location" (P2). Sometimes privacy policies are limited by protection mechanisms similar to the one we previously mentioned. Yet there would be no problem to implement P1 and P2 in such a mechanisms, the former simply restricts the audience of some piece of information, the latter defines the set of users which are allowed to perform an action. However, too restrictive privacy policies reduce the amount of information which is shared in the SNS. It makes the SNS less attractive, which can reduce the amount of users or decrease its growth. For this reason SNSs try to provide privacy protection mechanisms, which do not excessively limit users interaction.

Consequences of activating a privacy policy

Many users are not fully aware of the result of activating a privacy policy or if the policy protects their resources as they expect. Privacy concerns users often take the so called "friend-only" approach, by which they expect their resources to be shared only with their friends. Most SNSs require the friendship connection to be a consensus between the two users involved. Due to this, users tend to feel comfortable sharing with their friends, since they are more or less aware of who they are. Unfortunately, the friend-only approach can be easily bypassed. Consider a Facebook user who sets the default audience of her pictures to only her friends. In Facebook when a user is tagged on a picture, the audience of this picture is always extended with the friends of the tagged user, which shows that the friend-only strategy can be easily infringed simply by tagging. This is just one example, but there are many actions which have implicit consequences; e.g. when users join an event they disclose their potential location during the event to all the guests, or when commenting on a post the audience of this comment is the same than the post's audience. According to the aforementioned studies, users are not fully aware of these implicit consequences. Nevertheless they cannot be blamed for not understanding all possible results of an action. Instead the privacy mechanism should be responsible for blocking any event which discloses the resources of a user. For instance, if the user defines the audience of a picture to be her friends, tagging should not automatically extend the audience of the picture, or maybe the tagging action should be forbidden.

All in all, these two privacy issues show that privacy policies should be more expressive and their effects must be easy to understand. Users should immediately know the outcome of activating any privacy policy. SNSs put more and more effort in improving their privacy mechanisms and offering users better control over their information, yet, the increasing amount of personal data that SNSs have to deal with and the continuous changes in the privacy policies make this task cumbersome and hard to accomplish.

1.3 Thesis overview

Our aim in this thesis is to provide a suitable formalism for writing and reasoning about privacy policies in SNSs, and to enable a formal assessment on whether these policies are preserved as the SNS evolves. 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), together with a conformance relation to be able to state whether a certain social network satisfies a given policy.

Epistemic logic has been successfully used as a formal specification language to reason about security protocols [30], information security [19] and more generally about properties of distributed systems [15]. Here we propose to use an epistemic framework for privacy in SNSs. We start with first-order logic to represent connections and relationships between users and enrich it with epistemic (K) and deontic (P) operators to express knowledge and permissions, respectively. For instance, the logical formula $P_{Bob}^{Alice} tag$ means that "Bob is permitted to tag Alice in any picture", hence we can write $[\![\neg P_{other}^{me} tag]\!]_{me}$ to model the policy P1 we wrote above, that said "I do not want be tagged in pictures by anyone other than myself". The wrapper $[\![\;]\!]_{me}$ is used to specify the owner of the privacy policy, in this case me. Similarly we can write $S_{All}\, location(me)$, which stands for "Someone among all users in the SNS knows my location". Now we can formalise the privacy policy "Nobody apart from myself can know my child's location", which corresponds to P2, as $[\![\neg S_{All \setminus \{me\}} location(myChild)]\!]_{me}$, and disallow any user apart from the owner of the privacy policy to know the child's location. In the logic it is possible to nest knowledge operators in order to express more precise privacy polices. 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: "$Alice$ does not know that Bob knows about the event ev".

We use the reasoning machinery and the logical formalisation to cope with dynamic evolution of SNSs. As we mentioned before, some actions can implicitly extend the audience of a resource. In our formalism we capture the behaviour of all possible actions that users can execute in the SNS. Because of this, we are able to detect any violation of any privacy policy activated by the user. This is done by introducing generic operational semantics rules which capture the dynamic features of the SNS. In this way, if tagging extends the audience of a picture, the operational semantics rules would describe it and if the user defined the audience of her pictures to be her friends, the execution of the tagging action would lead to a violation of the privacy policy. It provides a more effective privacy protection, since users do not have to be aware of all events which could violate their privacy policies. They only need to specify who is able to access the information and it is protected against any action which violates the policy. When the privacy breach is triggered the SNS could react in different ways, e.g. blocking the tagging, or allowing it but not extending the audience. The rules are divided in four categories depending on how the knowledge, the permissions, the social graph topology, the policies or a combination of them is updated as the agents perform actions.

We use our framework to study the privacy policies of Twitter and Facebook and to asset the impact of adding new desirable ones. We start off by providing instantiations of the framework for Twitter and Facebook. Then we prove that Twitter is privacy preserving, according to our formal definition of privacy-preservation, which in short, guarantees that none of the actions that users can execute would violate any privacy policy. Additionally we consider privacy policies that are not present neither in Facebook nor in Twitter, and show that they are not privacy preserving under these desirable privacy policies.

Contributions

Concretely, the contributions in this thesis are:

1. A first-order framework for defining and reasoning about privacy policies, having the following features: i) A social network model (SNM) which encapsulates the knowledge, permission and connections present in SNSs; ii) A knowledge-based logic is defined with the usual epistemic modalities plus a special permission operator; iii) A formal language for defining privacy policies which is very expressive, having among its capabilities the possibility to write policies containing nested knowledge.
2. Operational semantics rules describing the different ways SNSs may evolve. We consider four different types of such rules depending on what is changed : i) *Epistemic*, concerned with changes on the knowledge of (some) users; ii) *Topological*, concerned with changes in the structure of the network (graph); iii) *Policy*, where what is affected are the privacy policies themselves; iv) *Hybrid*, when what is changed is a combination of some of the above 3 types of rules. We provide specific rules for describing how Twitter works.
3. A proof that Twitter is privacy-preserving with respect to all its possible events, and the set of privacy polices offered

today by the SNS .

4. A proof that Twitter and Facebook are not privacy-preserving after extending their privacy polices with some additional policies. Also, a proof that Facebook is privacy-preserving after modifying its operational semantics rules accordingly.

Outline

We start by introducing the static part of the first-order privacy policy framework in Chapter 2, where we also define the (static) instantiations of Twitter. In Chapter 3, we extend the framework with generic operational semantics rules which are used to describe the dynamic behaviour of SNSs. We also extend the mentioned instantiation with its dynamics. We formally define what a privacy-preserving SNS is in Chapter 4. Moreover we prove that Twitter is privacy-preserving and also that Twitter and Facebook would not preserve privacy after adding new policies to the ones they already support. The relation of our formalism to traditional epistemic logic is also discussed in Chapter 5. We conclude by discussing models for knowledge evolution and another protection mechanism for SNSs, and presenting our conclusions and future work in Chapter 6.

Statement of Contributions

This thesis comprises two papers:

- *A Formal Privacy Policy Framework for Social Networks* [28]. In this paper the static part of $\mathcal{FPPF}$ (called $\mathcal{PPF}$) is presented (Chapter 2), but everything was built using propositional logic. This paper was co-authored by Gerardo Schneider. Together we came up with social network models which keep the original structure of the social network and explicitly represent the knowledge and permission of the users. I developed the theory about social networks models, the knowledge-based language $\mathcal{KBL}_{\mathcal{SN}}$ and the privacy policy language $\mathcal{PPL}_{\mathcal{SN}}$. Moreover, I wrote the instantiation of Facebook and Twitter and all their privacy policies in $\mathcal{PPF}$. This paper was published in the proceedings of the *12th edition of the International Conference on Software Engineering and*

Formal Methods (SEFM'14).

- *A Formal Approach to Preserving Privacy in Social Networks* [27]. This technical report was co-authored by Musard Balliu and Gerardo Schneider. I developed all the theory regarding the dynamics of $\mathcal{FPPF}$ (Chapter 3) and the concept of privacy-preserving SNS. I instantiated Twitter and Facebook and wrote the proofs of privacy-preservation in both SNSs. Concretely, Theorem 1 and Lemmas 1, 2 (Chapter 4).

This thesis is an extended version of [27]. In particular, I have included more examples for some definitions; the proof of Lemma 3 and the new Facebook's operational semantics rules which made our model of the SNS privacy-preserving (Chapter 4); and an extended discussion which states the differences between traditional epistemic logic and our formalism (Chapter 5) where I informally describe part of the results we are currently investigating about this issue.

Chapter 2

Privacy Policy Framework

In this Chapter 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 to reason about privacy policy requirements of the users in a precise and formal manner. The framework is initially defined for generic social networks, but not all SNSs have the same particularities. Due to this, we also introduce the concept of *instantiations*, and we show how to instantiate Twitter.

2.1 The First-Order Privacy Policy Framework

The first-order privacy policy framework is equipped with several components. Firstly, we define models which leverage the well-known model for SNSs, social graph [14]. These models are enriched with the knowledge and the permission that users have in the SNSs. The knowledge is represented using a first-order epistemic (knowledge-based) structure, very much in the style of interpreted systems [15], and the permissions are represented as links between users in the graph, similarly to connections. Secondly, a knowledge-based logic is introduced to reason about all the properties that the models contain. Finally, the framework has an expressive language to write privacy policies, which is based on the aforementioned logic. Formally the framework is defined as follows:

Definition 1 (First-Order Privacy Policy Framework)**.** *The tuple*

$\langle \mathcal{SN}, \mathcal{KBL}_{\mathcal{SN}}, \models, \mathcal{PPL}_{\mathcal{SN}}, \models_C \rangle$ *is a* first-order privacy policy framework *(denoted by $\mathcal{FPPF}$), where*

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

In what follows we provide a more detailed description of each of the components in Def. 1.

2.1.1 Social Network Models

As we mentioned in the Chapter 1, social networks are usually modelled as graphs, where nodes represent the users of the SNS, and edges represent different kinds of relationships among agents, for instance information about their sentiments or any other social network specific information. These graphs are traditionally denoted as social graphs [14]. Users in social graphs, will be called *agents* in our models, and in the rest of the paper we will indistinguishably refer to them using either of the previous terms. Therefore, the agents in our models will also be represented as nodes. We store the knowledge that the agents have, which is represented as a collection of formulae in the knowledge-based language. Moreover, we model possible inferences of knowledge that the agents can perform from the knowledge that they already possess. Since we preserve the social graph structure, the edges of our models have the information about the relationships among users. Additionally, we use new types of edges to represent the permission that the agents have. Formally,

Definition 2. *Given a set of formulae $\mathcal{F}$, a set of privacy policies Π, and a finite set of agents $Ag \subseteq \mathcal{AU}$ from a universe $\mathcal{AU}$, then 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 (relational) structure over the social network model. As usual it consists of a set of domains $\{D_i\}_{i\in\mathcal{D}}$, a set of relations R_i, functions f_i and constants c interpreted over the domain.*

- *$KB : Ag \to 2^{\mathcal{F}}$ is a function giving 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 $KB(i)$.*

- *$\pi : Ag \to 2^{\Pi}$ is a function returning the set of privacy policies of each agent. We write π_i for $\pi(i)$.* $\square$

In the previous definition 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. Hereafter we use $\mathcal{C}, \Sigma$ and $\mathcal{D}$ to denote sets of indexes for connections, permissions and domains, respectively. Sometimes, we write an atomic predicate, 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.

As we mentioned, we want to provide agents with a reasoning capabilities which allow them to infer new knowledge. For instance, we would like Alice to be able to infer Bob's location, given that she knows that Bob is going to an event. In other words, if Alice knows that Bob is going to an event, then Alice can infer Bob's location. Since the knowledge of the agents is represented using formulae written in a formal language similar to that of epistemic logic, we will use the properties of knowledge that have extensively been studied in such a logic.

Specifically, here we formally introduce the minimum set of axioms and rules with which an agent can infer new knowledge from the one present in her KB. We will use as a minimal knowledge axiomatisation the set of axioms **K** from first-order epistemic logic [15, 26]. The language for first-order epistemic logic, $\mathcal{L}_n$, is recursively defined as follows:

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

where i is an agent from a set of agents AGT and $p(\overrightarrow{t})$ denote an atomic predicate over terms $\overrightarrow{t}$, for now it is enough to assume that terms can be either a constant symbol or function symbol (with implicit arity) or a variable. We refer the reader to Chapter 5 for a formal definition of the syntax of $\mathcal{L}_n$ and its semantics. The intuitive reading for the formula $K_i\varphi$ is "agent i knows φ". Hence we can use atomic predicates to denote concrete pieces of information, e.g. Bob's location can be represented as $location(Bob)$ and the statement "Alice knows Bob's location" can be written as $K_{Alice}location(Bob)$.

Now we introduce the most simplistic set of properties of knowledge defined in epistemic logic, which is called **K** axiomatisation and it is formally defined as follows:

Definition 3 (First-Order **K** [15, 26]). *Given the formulae φ and ψ written in $\mathcal{L}_n$ and some agent i, the axiom system **K** 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) $\forall x_1, \cdots, x_k.K_i\varphi \implies K_i\forall x_1, \cdots, x_k.\varphi$

Derivation rules

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

(Necessitation) $\dfrac{\varphi}{K_i\varphi}$

(Generalisation) $\dfrac{\varphi}{\forall x.\varphi}$

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

Definition 4 ([26]). *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 or A12) 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,* $\Gamma = \psi_1 \wedge \psi_2 \wedge \ldots \wedge \psi_n$ *we write* $\Gamma \vdash \varphi$*.*

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

$$K_{Alice}(going(u, \eta) \implies location(u, \eta)) \tag{2.1}$$

for any $u \in AGT$ and $\eta \in \mathbb{N}$, hence if Alice knows that Bob is going to the event η,

$$K_{Alice}going(Bob, \eta) \tag{2.2}$$

she can apply the axiom (A2) together with (2.1) and (2.2) 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 **K**, which generates all the knowledge that an agent can infer given the set formulae representing the explicit knowledge that she already has, KB. It 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 **K**. Different axioms may hold when $\mathcal{FPPF}$ is instantiated with a concrete social network model. To this end we defined the minimal Cl using the axioms and derivation rules from **K** and, in order to provide the agents with more targeted deductive engines, we keep open to extension 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 2.2 we will show how Cl can be extended to meet the requirements of a given SNS.

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

Fig. 2.1 shows a graphical representation of the aforementioned SN*. In this model the dashed arrows represent connections. Note that the Friendship connection is bidirectional, i.e. Alice is friend with Bob and*

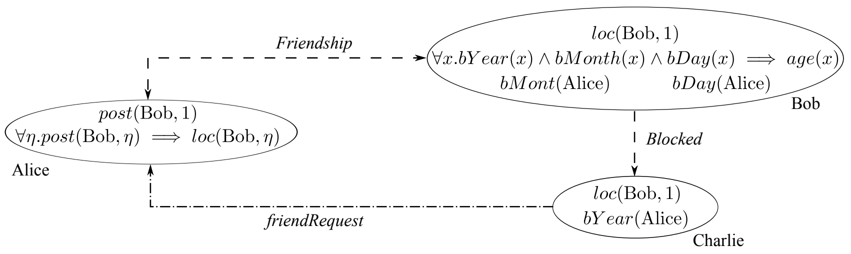
Friendship
$loc(\text{Bob},1)$
$\forall x.bYear(x) \wedge bMonth(x) \wedge bDay(x) \implies age(x)$
$bMont(\text{Alice})$ $bDay(\text{Alice})$
Bob
$post(\text{Bob},1)$
$\forall \eta.post(\text{Bob},\eta) \implies loc(\text{Bob},\eta)$
Alice
Blocked
$loc(\text{Bob},1)$
$bYear(\text{Alice})$
Charlie
friendRequest

Figure 2.1: Example of Social Network Model

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 and dashed arrow. In this example, Charlie is able to send a friend request to Alice.

The predicates inside each node represent the agents' knowledge. In this SNM, Charlie and Bob have the predicate $loc(Bob, 1)$ inside the node, meaning that both know location number 1 of Bob. However, note that there are, not only atomic predicates, but also formulae inside the agents' nodes. These formulae may increase the knowledge of the agents. For instance, Alice knows $loc(Bob, 1)$ implicitly. Since the deductive engine includes (among others) the rule Modus Ponens*, Alice can derive that, if she has access to a post of Bob, she can infer his location, i.e. $\forall \eta.post(Bob, \eta) \implies loc(Bob, \eta)$. Alice has access to $post(Bob, 1)$, therefore she can infer that $loc(Bob, 1)$.* □

2.1.2 The Knowledge-based Logic

We use the logic $\mathcal{KBL}_{\mathcal{SN}}$ 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 $i \in \mathcal{I}$ a family of functions with implicit arity. Then the terms are inductively defined as:*

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

where $\mathcal{I}$ is a set of indexes and $\overrightarrow{t}$ denotes a tuple of terms respecting the arity of f_i.

We use terms to define predicates. For instance, the predicate $friends(Alice, Bob)$ 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 $n \in \Sigma$, $G \subseteq Ag$ and $n \in \mathbb{N}$, the syntax of the* knowledge-based logic *$\mathcal{KBL}_{\mathcal{SN}}$ is inductively defined as:*

$$\begin{aligned}\varphi \ ::= \ & p(\overrightarrow{t}) \mid c_m(i,j) \mid a_n(i,j) \mid \varphi \wedge \varphi \mid \neg\varphi \mid \forall x.\varphi \mid \\ & K_i\varphi \mid E_G\varphi \mid S_G\varphi \mid D_G\varphi \mid C^n_G\varphi\end{aligned}$$

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 $\mathcal{KBL}_{\mathcal{SN}}$ 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. $C^n_G\varphi$, φ is the n-bounded common knowledge in the group G. We use the following operator as syntactic sugar in the logic $\mathcal{KBL}_{\mathcal{SN}}$: $P^j_i a_n := a_n(i,j)$, agent i is permitted to execute action a_n to agent j; $SP^j_G a_n := \bigvee_{i\in G} a_n(i,j)$, at least one agent in G is permitted to execute action a to agent j; $GP^j_G a_n := \bigwedge_{i\in G} a_n(i,j)$, 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. We define E^{k+1}_G as $E_G E^k_G\varphi$, where $E^0_G\varphi$ is equal to φ.

In what follows we define the satisfaction relation for $\mathcal{KBL}_{\mathcal{SN}}$ formulae, interpreted over social network models and agents. In particular, predicates are interpreted as relations over the respective domains in the model. Moreover, $\varphi[v/x]$ denotes the usual capture-free substitution in first-order logic and we tacitly assume that each variable v is mapped to its own domain.

$SN, u \models p(\vec{t})$	iff	$p(\vec{t}) \in Cl(KB_u)$
$SN, u \models \neg\varphi$	iff	$SN, u \not\models \varphi$
$SN, u \models \varphi \wedge \psi$	iff	$SN, u \models \varphi$ and $SN, u \models \psi$
$SN, u \models \forall x.\varphi$	iff	for all $v \in D_o$, $SN, u \models \varphi[v/x]$
$SN, u \models K_i\varphi$	iff	$\varphi \in Cl(KB_i)$
$SN, u \models c_m(i, j)$	iff	$(i, j) \in C_m$
$SN, u \models a_n(i, j)$	iff	$(i, j) \in A_n$
$SN, u \models S_G\varphi$	iff	there exits $i \in G$ such that $SN, u \models K_i\varphi$
$SN, u \models E_G\varphi$	iff	$SN, u \models K_i\varphi$ for all $i \in G$
$SN, u \models C^k_G\varphi$	iff	$SN, u \models E^n_G\varphi$ for $n = 1, 2, \ldots, k$
$SN, u \models D_G\varphi$	iff	$\varphi \in Cl(\bigcup_{i \in G} KB_i)$

Table 2.1: $\mathcal{KBL}_{\mathcal{SN}}$ satisfiability relation

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 finite set of agents $G \subseteq Ag$, $m \in \mathcal{C}$, $n \in \Sigma$, $o \in \mathcal{D}$ and $k \in \mathbb{N}$, the* satisfiability relation $\models \subseteq SN \times Ag \times \mathcal{KBL}_{\mathcal{SN}}$ *is defined as shown in Table 5.1.* □

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 the agent explicitly knows φ, i.e. φ is in her knowledge base, KB_i, or it can be derived (using the axiomatisation **K**) from the already existing formulae in her knowledge base ($\varphi \in Cl(KB_i)$). This definition is better illustrated by an example.

Example 2. *Let SN be the SNM in Fig. 2.1. As we described in Example 1, Alice knows post 1 of Bob, meaning that*

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

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

$$post(Bob, 1) \in KB_{Alice}. \qquad (2.3)$$

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

$$SN, Alice \models K_{Alice} loc(Bob, 1) \tag{2.4}$$

should hold. According to the semantics we have provided for K_i, the previous statement is true iff $loc(Bob, 1) \in Cl(KB_{Alice})$. Fig. 2.1 shows that, in SN, the following formula is in KB_{Alice}

$$\forall \eta . post(Bob, \eta) \implies loc(Bob, \eta) \tag{2.5}$$

where $\eta \in \mathbb{N}$[1]*, hence*

$$post(Bob, 1) \implies loc(Bob, 1) \tag{2.6}$$

is also in KB_{Alice}. From (2.3) and (2.6) 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 (Def. 5), modus ponens *can by applied for (2.6) and (2.3) to derive $loc(Bob, 1)$, i.e. $loc(Bob, 1) \in Cl(KB_{Alice})$ and therefore (2.4) holds.* $\square$

Note that the interpretation of atomic predicates $p(\overrightarrow{t})$ is similar to that of $K_i\varphi$. There are cases when the evaluation of a predicate in a node is equivalent to a formula containing the knowledge modality. For instance,

$$SN, i \models p(\overrightarrow{t}) \equiv SN, i \models K_i p(\overrightarrow{i})$$

by definition of $\models$, both statements are true iff $p(\overrightarrow{t}) \in Cl(KB_i)$. However, it does not hold in general for a formula φ. Consider $\varphi := \neg p(\overrightarrow{t})$, then

$$SN, i \models \neg p(\overrightarrow{t}) \not\equiv SN, i \models K_i \neg p(\overrightarrow{i})$$

since the left hand side is true iff $p(\overrightarrow{t}) \notin Cl(KB_i)$ and the right hand side holds iff $\neg p(\overrightarrow{t}) \in Cl(KB_i)$.

The interpretation of distributed knowledge, D_G, is also similar to the one for K_i, but considering the knowledge of all agents in G instead of taking into account only the knowledge of agent i.

[1] We tacitly assume that variables are mapped to their respective domains.

In the next section, Example 6 shows the evaluation of a formula involving distributed knowledge.

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

Example 3. *Consider again the SNM in Fig. 2.1, denoted as SN. We can now check whether or not the following expression*

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

holds for $i \in \{Alice, Bob, Charlie\}$. As we mentioned 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

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

it implies that

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

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

$$SN, i \models P^{Alice}_{Charlie}friendRequest.$$

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

Not all SNSs are characterised by the same knowledge and permission properties. Different properties hold in different SNSs. As we have seen, using the satisfiability relation $\models$, we can whether a $\mathcal{KBL}_{\mathcal{SN}}$ formula holds in a SNM. These knowledge and permission properties can be also expressed in $\mathcal{KBL}_{\mathcal{SN}}$, and consequently, we can check whether they hold in 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:*

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

□

Relation to Classical Epistemic Logic. The deductive engine allows the agents to apply the axioms and derivation rules in **K** to derive new knowledge. The same axiomatisation cannot be used for $\mathcal{KBL}_{\mathcal{SN}}$, since connection and permission predicates are interpreted differently depending on whether they are inside a knowledge modality. For instance, satisfiability 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 unconventional interpretation of some predicates prevents us from using axiomatisations defined for classical epistemic logic. However, when checking if a formula is in the KB 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 $\mathcal{L}_n$ and, because of this, we assume that they can infer new knowledge using the axiomatisation **K**.

To sump up, we can think of SNMs as models that combine two logics. On one hand, $\mathcal{KBL}_{\mathcal{SN}}$ is used to reason about the global knowledge and permission of the SNS. On the other hand, agents have their knowledge represented using $\mathcal{L}_n$ and they use **K** to infer new knowledge. The relation of $\mathcal{KBL}_{\mathcal{SN}}$ to the traditional Kripke models in discussed in deeper detail in Chapter 5.

2.1.3 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 $\mathcal{PPL}_{\mathcal{SN}}$ as a formal language for writing privacy policies based on $\mathcal{KBL}_{\mathcal{SN}}$.

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

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

In $\mathcal{PPL}_{\mathcal{SN}}$ 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 category α represents the restrictions which 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 the 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 $\mathcal{PPL}_{\mathcal{SN}}$ 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 $\mathcal{PPL}_{\mathcal{SN}}$ agent i can write $[\![\neg S_{\{m,n,o\}} p(\overrightarrow{t})]\!]_i$, meaning that none of the agents $m, n, o \in Ag$ can know $p(\overrightarrow{t})$.
- *Conditional restrictions* - $[\![\varphi \implies \neg\alpha]\!]_i$ A restriction α is enforced depending on some knowledge or permission state.

Example 5. *As an example consider the following policy:*

$$\forall j.[\![\neg P^i_j join_{event(i)} \implies \neg K_j event(i, descp)]\!]_i \tag{2.7}$$

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 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 $\mathcal{PPL}_{\mathcal{SN}}$ as

$$[\![\neg S_{Ag \setminus friends(i)} event(i, descp)]\!]_i \tag{2.8}$$

Unlike (2.7), this policy is enforced in most SNSs. However (2.8) is much more coarse-grained than (2.7) and, as a result, it will not allow some users to access the event if they are able to join it. Consequently, (2.8) unnecessarily reduces the audience of the event.

Example 6. *The distributed knowledge operator D_G allows to protect users' against intricate leaks of information in groups of agents. Consider the social network model presented in Fig. 2.1, where Bob knows the day and the month of Alice's birthday, denoted by $bDay(\textit{Alice})$ and $bMonth(\textit{Alice})$, respectively and he can also infer the age of a user whenever he knows the user's full date of birth, i.e.,*

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

Moreover, Charlie knows the year of Alice's birth, represented by the predicate $bYear(\textit{Alice})$. Therefore, if Bob and Charlie combine their knowledge, Alice's age, $age(\textit{Alice})$, 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 $\mathcal{PPL}_{\mathcal{SN}}$ Alice can write the privacy policy

$$[\![\neg D_{\{Bob,Charlie\}} age(a)]\!]_{Alice}$$

to prevent this leak. Note that the social network model considered in this example violates the policy. □

The previous examples show that the privacy policies that we can express in $\mathcal{PPL}_{\mathcal{SN}}$ give users a more 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 $SN = \langle Ag, \mathcal{A}, KB, \pi \rangle$, 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 *$\models_C$ is defined as shown in Table 2.2.* □

Note that $\models_C$, is defined using the satisfiability relation $\models$. 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 (2.7) from Example 5 and a random SNM SN

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

$SN \models_C \delta_1 \wedge \delta_2$	iff	$SN \models_C \delta_1 \wedge SN \models_C \delta_2$
$SN \models_C \forall x.\delta$	iff	for all $x \in D_o$, $SN \models_C \delta[v/x]$
$SN \models_C [\![\neg\alpha]\!]_i$	iff	$SN, i \models \neg\alpha$
$SN \models_C [\![\varphi \implies \neg\alpha]\!]_i$	iff	$SN, i \models \varphi$ then $SN \models_C [\![\neg\alpha]\!]_i$

Table 2.2: $\mathcal{PPL}_{\mathcal{SN}}$ conformance relation

By applying the semantics defined in Table 2.2, checking whether SN is in conformance with the policy is equivalent to check that for all $u \in Ag$

$$SN \models_C [\![\neg P_u^i join_{event(i)} \implies \neg K_u event(i, descp)]\!]_i$$

which is also equivalent to

$$\text{If } SN, i \models \neg P_u^i coin_{event(i)} \text{ then } SN, i \models \neg K_u event(i, descp)$$

As you can see in Table 2.2, checking conformance of any formula in $\mathcal{PPL}_{\mathcal{SN}}$ boils down to checking satisfiability of the corresponding formula in $\mathcal{KBL}_{\mathcal{SN}}$.

2.2 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 the users has been. In Google+ users are group 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 not present in other SNSs Foursquare, Twitter or Instagram.

Here we introduce the concept of $\mathcal{FPPF}$ instantiation, which will be used to model all the specific characteristics of each SNSs. We formally define an instantiation for an SNS as follows:

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

$$\mathcal{FPPF}_{\mathcal{S}} = \langle \mathcal{SN}_{\mathcal{S}}, \mathcal{KBL}_{\mathcal{SN}}, \models, \mathcal{PPL}_{\mathcal{SN}}, \models_C \rangle$$

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 atomic predicates $\mathcal{P}_{\mathcal{S}}$, a set of possible connection relations $\mathcal{C}_{\mathcal{S}}$, a set of permission relations $\Sigma_{\mathcal{S}}$, 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 $\mathcal{KBL}_{\mathcal{SN}}$ (these properties are 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 $\mathbb{AC}_{\mathcal{S}}$.*

The *admissibility* condition $\mathbb{AC}_{\mathcal{S}}$ specifies the *generic structure* of privacy policies of a particular instantiation (see Def. 12 for an example). Formally, $\mathbb{AC}_{\mathcal{S}}$ is a predicate over the elements of $\mathcal{FPPF}_{\mathcal{S}}$ defining the well-formed policies for the instantiation. We write $\pi' \in \mathbb{AC}_{\mathcal{S}}$ if π' satisfies $\mathbb{AC}_{\mathcal{S}}$. 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 instantiations.

2.2.1 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 a user wants to re-share a tweet, she can use the *retweet* functionality which shares an already published tweet from another user. Users can also *favourite* tweets, which is similar to star emails, i.e. it marks the tweet with a start. 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 a user follows another user, she gets updates with all the tweets of the users she follows.

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

Atomic predicates. Given $o, u \in Ag$ and $\mu, \eta \in \mathbb{N}$, the set of atomic 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 favourited $tweet(o, \eta)$.
- $retweet(u, o, \eta)$ - 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:

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

where $accessProf$ is the action of a user accessing other user's profile; $accessProfRec$ represents the fact 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; $sendAd$ is the action of an advertisement company sending advertisements to a user.

Constraints over privacy policies. 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 [28], which was shown to express all possible policies of Twitter 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.

Auxiliary functions. The set of auxiliary functions consists of:

- $followers : Ag_{\textit{Twitter}} \rightarrow 2^{Ag_{\textit{Twitter}}}$ - This function returns all the followers of a given user, i.e. for $u \in Ag_{\text{Twitter}}$, $followers(u) = \{\, i \mid iC_{Follower}u \,\}$.
- $state : Ag_{\textit{Twitter}} \rightarrow St$ - 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.
- $inclocation : Ag_{\textit{Twitter}} \rightarrow Bool$ - This function returns the user 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.
- $beingReco : Ag_{\textit{Twitter}} \rightarrow Bool$ - 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.
- $getTweetInfo : Ag_{\textit{Twitter}} \times \mathbb{N} \rightarrow 2^{\mathcal{P}_{\textit{Twitter}}}$ - 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 atomic 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 $p(\vec{t}) \in \mathcal{P}_{\text{Twitter}}$, $audience(p(\vec{t})) = \{\, i \mid SN, i \models K_i p(\vec{t})\}$
- $info : Ag_{Twitter} \rightarrow 2^{\mathcal{P}_{Twitter}}$ - This function returns all the information of a given agent. Given an agent $u \in Ag_{\text{Twitter}}$, $info(u) = \{p(u, \vec{t}) | p(u, \vec{t}) \in KB_u\}$.

Properties of $\mathcal{FPPF}_{\text{Twitter}}$. The role of the assumptions is twofold. Firstly, they are used to encode some of the properties of the SNS and, secondly, some of these assumptions are added to the knowledge base KB of the agents. Note that in the following set of properties, we write that an agent has access to a predicate $p(\vec{t})$ when she knows it, i.e. $K_i p(\vec{t})$. 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 $p(\vec{t}) \in oracle(o, \eta)$,

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

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

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

  It 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 profile.

$$\begin{aligned}&\forall i.\forall u.\forall o.\forall \eta.(K_i retweet(u, o, \eta) \implies \\ &\quad P_i^u accessProf \;\wedge P_i^o accessProf)\end{aligned}$$

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

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

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 gets 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 $profile(u)$, which as soon as it is learnt by a user, it permits her 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 4 properties introduced here.

Summary of the Chapter

In this chapter we have presented the first-order privacy policy framework $\mathcal{FPPF}$. We have also introduced the notion of instantiation of the framework and we have shown how Twitter, one of the most popular SNSs nowadays, can be instantiated. So far, we are only able to represent a static picture of the SNS. We are not considering how SNMs change as events occur. In the next chapter, we describe the dynamics of $\mathcal{FPPF}$ and we show how to extend the Twitter instantiation with its dynamic behaviour.

Chapter 3

Privacy Policies in Evolving SNS

Social network users are able to execute events. For example, they can post messages on a timeline, they can like a given post, share pictures and so forth. Different events will change the knowledge and the permission of the SNS in different ways. In this chapter, we formally incorporate the events that can be executed in the SNS and the operational semantics rules modelling the events' behaviour in $\mathcal{FPPF}$. These rules formally describe how SNMs change when a particular event occur. It 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. Similarly to in the previous chapter, we describe how these elements are instantiated for particular social networks and we conclude the chapter by extending the Twitter instantiation that we provided in Chapter 2.

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 change of state.

In order to represent the behaviour induced by the events of the SNS, we define an LTS, which is used 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. Edges indicate, which is the resulting configuration (or SNM) after executing an event in a given configuration. In what follows we formally introduce the SNS LTS.

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

- *$Conf$ is a (finite) set of all possible 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 $\mathcal{FPPF}$ evolving SNSs as described by the Labelled Transition System.

Definition 14 (Dynamic First-Order Privacy Policy Framework). *The tuple $\langle \mathcal{LTS}_{\mathcal{SN}}, \mathcal{KBL}_{\mathcal{SN}}, \models, \mathcal{PPL}_{\mathcal{SN}}, \models_C \rangle$ 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;*
- *$\mathcal{KBL}_{\mathcal{SN}}$ is a knowledge-based logic;*

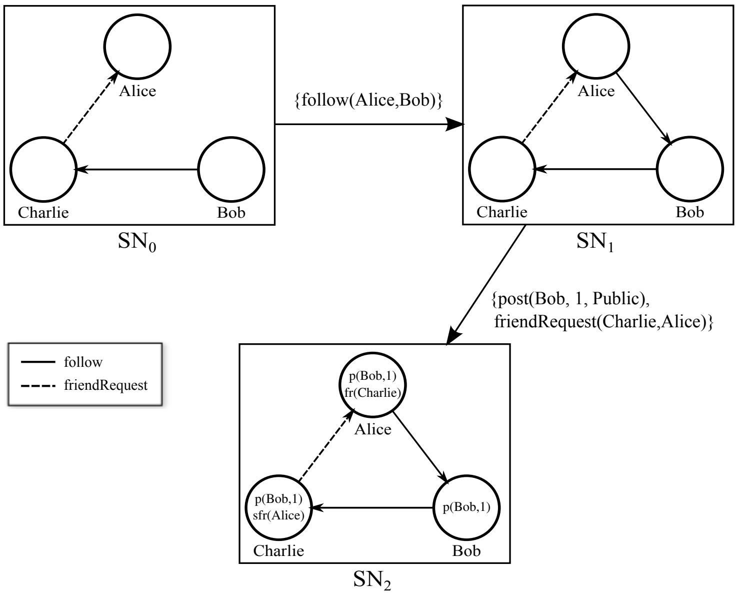

Figure 3.1: Example of SNS Labelled Transition System

- $\models$ *is a satisfaction relation defined for* $\mathcal{KBL}_{\mathcal{SN}}$*;*
- $\mathcal{PPL}_{\mathcal{SN}}$ *is a formal language for writing privacy policies;*
- $\models_C$ *is a conformance relation defined for* $\mathcal{PPL}_{\mathcal{SN}}$*.* □

Example 7. *In Fig. 3.1 we show an example of an SNSLTS. The rectangles represent 3 configurations* $SN_0, SN_1, SN_2 \in Conf$*. This SNSLTS shows one possible sequence of events that can be executed. Each configuration constitutes 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.*

There is a transition from SN_0 *to* SN_1*, representing that the SNS can evolve from its configuration* SN_0 *to* SN_1 *by the execution 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, as* *expected, with a directed arrow between them in the resulting SNM, SN_1. This transition is composed by only one event, which means that no other event was executed in parallel to it. As we mentioned before, transitions in SNSLTSs are labelled with sets of events representing the actions executed in parallel. In $SN_1 \xrightarrow{\{post(Bob,1,Public),friendRequest(Charlie,Bob)\}} SN_2$ 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 an atomic 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 it is needed to check whether a friend request can be sent. Charlie is allowed to perform this event since*

$$SN_1, Charlie \models P^{Alice}_{Charlie} friendRequest$$

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. It is not possible to formally describe the behaviour of each of the events. In the following sections we will introduce dynamic instantiation of our framework and the operational semantics rules which formally define the behaviour of each event.

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

As in the static case, the dynamic instantiation of an SNS requires to specify 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 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:*

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

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

3.2.1 Operational Semantics Rules for SNS

The dynamic behaviour of an SNS is described in terms of the 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 other auxiliary information which are used to describe the rule. They are defined by leveraging all the elements of $\mathcal{PPF}$ and the instantiation $\mathcal{FPPF}_\mathcal{S}$ in which the rules are defined. The operational semantics rules are divided in 4 types, as reported in Table 3.1. We use the superindex e whenever an update of the $\mathcal{SN}$ depends on event e. In what follows we first describe the intuition for each type of rules and then we provide a detailed description of the epistemic rule.

Epistemic: These rules are used to specify events which 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 (cf. the first premise of the epistemic rule in Table 3.1). 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.

Epistemic

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

Topological

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

Policy

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

Hybrid

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

Table 3.1: Generic structure of the operational semantics rules

Topological: These rules only affect the social topology of SNMs. The social topology represents the elements of an SNM which come from the social graph. Therefore these rules update the elements Ag and C_i of an SNM. Using topological rules we can model the addition or removal of users and 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 the SNS requires a mix of any of the previous types of rules. 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 3.1, here we provide a detailed explanation of the generic epistemic rule. Consider the first premise in the rule,

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

it 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 $Ag = \{\text{Alice}, \text{Bob}\}$, then for an event e_1 that only updates Bob's knowledge with the predicate $p(\vec{t})$ we would have $\Gamma^{e_1}_{Alice} = \emptyset$ and $\Gamma^{e_1}_{Bob} = \{p(\vec{t})\}$.

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 $i \in \Sigma$[1] 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 permission, 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

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

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

[1]Remember that in SNMs we use binary relations between agents to represent permission (cf. Section 2.1.1)

How does knowledge spread out in an SNS?

The usual way of releasing information in SNSs is making available a message to a group of users, e.g. posts on Facebook, tweets on Twitter, pictures on Pinterest, tips on Foursquare, etc. In dynamic epistemic logic this type of event is known as *public announcement* [10]. The result of performing a public announcement is that the disclosed information becomes *common knowledge* in the group of agents which are the audience of the announcement. In the traditional epistemic logic semantics [15], common knowledge is modelled as an infinite chain of nested knowledge over a model M and a state s,

$$(M, s) \models C_G\varphi \text{ iff } (M, s) \models E^n_G\varphi \text{ for } n = 0, 1, 2, \ldots$$

This representation of common knowledge accurately models the intuitive notion of what is known for everyone in a group. The explicit representation of knowledge is not suitable for the classical semantics of the common knowledge operator. In Chapter 2 we introduced the *k-bounded common knowledge* modality, which is a restricted representation of the same concept. k-bounded common knowledge not only enables the possibility of representing and reasoning about common knowledge in $\mathcal{FPPF}$, but it also provides flexibility when modelling the amount of knowledge the agents gain after being part of a public announcement. The bounded common knowledge does not reduce the power of $\mathcal{FPPF}$ to detect privacy leaks. The amount of nested knowledge which the agents can have will be bounded by the deductive engine they are provided with. Therefore one can define which will be maximum nested knowledge that will appear in a social network model and choose a precise boundary k.

For instance, in $\mathcal{FPPF}_{\text{Twitter}}$ (Section 3.2.1), we chose *3-bounded common knowledge*. This is because, firstly, the minimum set of axioms in the deductive engine does not increment the nested knowledge of the agents; secondly, the new axioms defined in $\mathbb{A}_{\text{Twitter}}$ do not increment the agents' nested knowledge either; and thirdly, because the set of privacy policies that the agents can write in π_{Twitter} do not contain nested knowledge. Consequently, a value greater than 3 would not change the result of the privacy analysis we carry out in the dynamic instantiation of Twitter. A bound less than 3 would not capture precisely what happens when *tweeting*, since after this events occurs, the tweet's audience knows that the owner of the tweet knows that they know the tweet, i.e. $E_{Audience}K_{owner}E_{Audience}tweet(owner)$.

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.

3.3 Dynamic Instantiation of Twitter

In this section we present the dynamic instantiation of Twitter, $\mathcal{FPPF}^{\mathcal{D}}_{\text{Twitter}}$, by extending the instantiation $\mathcal{FPPF}_{\text{Twitter}}$, which was introduced in Section 2.2. EVT_{Twitter} is the set containing all 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 users can block other users. This pair represents the events of blocking and unblocking a user, respectively.

- $showReco$ - 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.

We instantiate the rules in Table 3.1 for each of the events described above. As a result, we obtain the operational semantics rules for a given social network. Here we describe the Twitter rules for the events $createProf$ and $tweet$ modelled in Table 3.2. For the full set of rules modelling Twitter semantics please refer to Sections A.2.1, A.2.2, A.2.3 and A.2.4 in the Appendix.

The event $createProf$ describes how the social network model changes when a new user joins the SNS, i.e. $SN \xrightarrow{createProf(u,InitInfo)} SN'$ for $SN, SN' \in \mathcal{SN}_{\text{Twitter}}$, $u \in Ag$ and $InitInfo \subseteq \mathcal{F}_{\mathcal{KBL}}$ (representing the initial set of information that users provide in Twitter). Rule $R5$ 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, $KB'_i = InitialInfo$, 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.

Tweet

$$\frac{\begin{array}{c} Au = followers(tu) \cup \{u\} \cup \{u \mid mention(u, tu, \eta) \in TweetInfo\} \\ state(tu) == \text{'Public'} \qquad Inclocation(u) == true \\ \forall \varphi \in TweetInfo, \forall i \in Au\ KB'_i = KB_i \cup \{C^3_{Au}\varphi\} \end{array}}{\langle \_, \_, KB, \_ \rangle \xrightarrow{tweet(tu, TweetInfo)} \langle \_, \_, KB', \_ \rangle}$$

$$\frac{\begin{array}{c} Au = followers(tu) \cup \{u\} \qquad state(tu) == \text{'Private'} \\ Inclocation(u) == false \qquad location(tu, \eta) \notin TweetInfo \\ \forall \varphi \in TweetInfo, \forall i \in Au\ KB'_i = KB_i \cup \{C^3_{Au}\varphi\} \end{array}}{\langle \_, \_, KB, \_ \rangle \xrightarrow{tweet(tu, TweetInfo)} \langle \_, \_, KB', \_ \rangle}$$

$$\frac{\begin{array}{c} Au = followers(tu) \cup \{u\} \cup \{u \mid mention(u, tu, \eta) \in TweetInfo\} \\ state(tu) == \text{'Public'} \\ Inclocation(u) == false \qquad location(tu, \eta) \notin TweetInfo \\ \forall \varphi \in TweetInfo, \forall i \in Au\ KB'_i = KB_i \cup \{C^3_{Au}\varphi\} \end{array}}{\langle \_, \_, KB, \_ \rangle \xrightarrow{tweet(tu, TweetInfo)} \langle \_, \_, KB', \_ \rangle}$$

$$\frac{\begin{array}{c} Au = followers(tu) \cup \{u\} \\ state(tu) == \text{'Private'} \qquad Inclocation(u) == true \\ \forall \varphi \in TweetInfo, \forall i \in Au\ KB'_i = KB_i \cup \{C^3_{Au}\varphi\} \end{array}}{\langle \_, \_, KB, \_ \rangle \xrightarrow{tweet(tu, TweetInfo)} \langle \_, \_, KB', \_ \rangle}$$

Create Profile

$$\frac{\begin{array}{c} u \notin Ag \qquad Ag' = Ag \cup \{u\} \qquad KB'_i = InitialInfo \\ \forall j \in Advertisers\ A'_{sendAd} = A_{sendAd} \cup \{(u, j)\} \\ A'_{accessProf} = A_{accessProf} \cup \{(u, u)\} \end{array}}{\langle Ag, \{\{A_i\}_{i \in \mathcal{I}_2}, \_\}, KB, \_ \rangle \xrightarrow{createProf(u, InitialInfo)} \langle Ag', \{\{A'_i\}_{i \in \mathcal{I}_2}, \_\}, KB', \_ \rangle}$$

Table 3.2: Create and Tweet rules for $\mathcal{FPPF}^{\mathcal{D}}_{\text{Twitter}}$

In general, an event may give rise to more than one operational semantics rules. $tweet$ is an example of such event. $tweet$'s behaviour is modelled in Table 3.2. 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 $SN \xrightarrow{tweet(u, TweetInfo)} SN'$ for $SN, SN' \in \mathcal{SN}_{\text{Twitter}}$, $u \in Ag$

and $TweetInfo \in 2^{\mathcal{P}_{\text{Twitter}}}$ (representing the information disclosed in the tweet, i.e. location of the tweet, mentions, pictures, etc). In the first line of $tweet$ we specify which 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, $followers(u) \cup \{u\} \cup \{v | mention(v, u, \eta) \in TweetInfo\}$. Otherwise, the audience is restricted to only her followers $followers(u) \cup \{u\}$. If the tweet location is deactivated, $inclLoc(u) == false$, then the rules contain one extra condition which explicitly requires that the location should not be part of the information disclosed in the tweet, $location(u, \eta) \notin TweetInfo$. As a result, all the formulae describing the tweet information become common knowledge among the agents of the audience, $\forall \varphi \in TweetInfo, \forall i \in Au\ K'_i = K_i \cup \{C^3_{Au}\varphi\}$.

The reader may wonder why the audience of a tweet is not any Twitter user 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 the 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 (which 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 Table A.2.1 in the Appendix.

Summary of the Chapter

In this chapter we have presented the dynamics of the first-order privacy policy framework $\mathcal{FPPF}$. It has been done by introducing an LTS which captures the dynamic behaviour of the SNS. Concretely, each event that can be executed in the SNS is modelled by operational semantics rules, which define how the SNM evolves. We have also defined how to instantiate the dynamics of an SNS and we have extended the instantiation of Twitter that we provided in the previous chapter with its dynamic behaviour. However, we have not discussed the consequences, regarding to privacy, of executing an event. In the next chapter, we will define what means for an SNS (modelled in $\mathcal{FPPF}$) to be privacy-preserving and we will analyse this property in instantiations of Twitter and Facebook.

Chapter 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 chapter, 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.

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 preserve 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 a 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:*

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

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

4.2 Privacy in Twitter

Using the dynamic instantiation of Twitter that we defined in the previous chapter, $\mathcal{FPPF}^{\mathcal{D}}_{\text{Twitter}}$, we show that the described events in EVT_{Twitter} and the proposed specification using the operational semantics rules are privacy-preserving (as defined in Def. 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 combination of events and privacy policies (i.e. ensuring that Def. 16 holds). Here we only show the case when $tweet$ is executed and $P1$ is activated. We follow the same strategy for the remaining cases (See Appendix B.1 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:

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

2. By contradiction, let us assume
2.1. $SN \models_C P1(u)$ and $SN \xrightarrow{e} SN'$
2.2. $SN' \not\models_C P1(u)$

3. By 2.2.
3.1. $SN' \not\models_C P1(u)$ [Def. $\models_C$]
3.2. $SN', u \models \neg\neg S_{Ag \setminus followers(u) \setminus \{u\}} tweet(u, \eta)$ [$\neg\neg_e$]

3.3. $SN', u \models S_{Ag \setminus followers(u) \setminus \{u\}} tweet(u, \eta)$

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

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

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

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

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 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 Chapter 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 $\mathcal{PPL}_{\mathcal{SN}}$ 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* $P6(i) \in \mathbb{AC}_{\mathit{Twitter}}$ *where* $i \in Ag_{\mathit{Twitter}}$.

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

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

If the result of executing the event in SN is SN', $SN \xrightarrow{e} SN'$, then by assumption we know that $SN \models_C P6(u)$, 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 $SN' \not\models_C P6(u)$. See Appendix B.2 for the detailed proof. $\square$

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 $P6$.

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 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 $\mathcal{FPPF}_{\text{Facebook}}$, which exclusively contains the required elements to model the tagging process.

Given $o, tge, tgr \in Ag_{\text{Facebook}}$ and $\eta \in \mathbb{N}$, the set of atomic predicates, $\mathcal{P}_{\text{Facebook}}$, 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, i.e. $\mathcal{C}_{\text{Facebook}} = \{C_{Friendship}\}$. The actions set Σ_{Facebook} only contains the action $removeTag_{tag(tge,tgr,accepter,\eta)}$, which defines which users have permission to remove the tag $tag(tge, tgr, accepter, \eta)$. Regarding to auxiliary functions we only include:

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

The previous elements constitute the static instantiation of Facebook, $\mathcal{FPPF}_{\text{Facebook}}$. In order to model the behaviour of the tagging event, we extend $\mathcal{FPPF}_{\text{Facebook}}$ with the operational semantics rules for the events $\{tag, acceptTagRequest\} \subseteq EVT_{\text{Facebook}}$ as specified in Table 4.1, which defines $\mathcal{FPPF}^{\mathcal{D}}_{\text{Facebook}}$. The intuition behind the operational semantics rules is as follows.

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

Tag - FR1

$$\text{FR1.1}\ \frac{\begin{array}{c} picture(o,\eta) \in KB(tgr) \\ KB'(o) = KB(o) \cup \{C^3_{\{o,tgr\}} tagRequest(tgr,tge,o,\eta)\} \\ KB'(tgr) = KB(tgr) \cup \{C^3_{\{o,tgr\}} tagRequest(tgr,tge,o,\eta)\} \end{array}}{SN \xrightarrow{tag(tgr,tge,picture(o,\eta))} SN'}$$

Accept tag request - FR2

$$\text{FR2.1}\ \frac{\begin{array}{c} Au = audience(picture(o,\eta)) \cup friends(tge) \\ a = removeTag_{tag(tge,tgr,o,\eta)} \\ acptr == o \quad tagRequest(tge,tgr,o,\eta) \in KB(acptr) \\ A'_a = A_a \cup \{(o,o),(o,tge)\} \\ \forall i \in Au\ KB'(i) = KB(i) \cup \{C^3_{Au} tag(tge,tgr,o,\eta)\} \end{array}}{SN \xrightarrow{acceptTagRequest(acptr,tge,tgr,picture(o,\eta))} SN'}$$

Table 4.1: Tagging - Operational Semantics of Facebook

the picture and it becomes common knowledge for both of them, $\forall i \in \{o, tgr\}$

$$KB'(i) = KB(i) \cup \{C^3_{\{o,tgr\}} tagRequest(tgr,tge,o,\eta)\}$$

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

$$(o,o),(o,tge) \in A'_{removeTag_{tag(tge,tgr,accepter,\eta)}}.$$

Also the tag is disclosed to the users part of the audience of the picture, thus becoming common knowledge among them. $\forall i \in$

$audience(picture(o, \eta))$

$$KB'(i) = KB(i) \cup \{C^3_{Au} tag(tge, tgr, o, \eta)\}.$$

Consider 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 $FP1(u)$ where $u \in Ag_{\text{Facebook}}$ and it is expressed in $\mathcal{PPL}_{\mathcal{SN}}$ as follows:

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

meaning that for all $u, t, o \in Ag_{\text{Facebook}}$ and picture $picture(o, \eta)$ 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 $FP1(u) \in \mathbb{AC}_{\textit{Facebook}}$ where $u \in Ag_{\textit{Facebook}}$.*

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 $\mathcal{FPPF}^{\mathcal{D}}_{\text{Facebook}}$ 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 $SN, SN' \in \mathcal{SN}_{\text{Facebook}}$ 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 $SN', o \models E_{Au} tag(tge, o, o, \eta)$, hence $SN' \not\models_C FP1(tge)$, which contradicts our claim $SN' \models_C FP1(tge)$ and therefore $\mathcal{FPPF}^{\mathcal{D}}_{\text{Facebook}}$ is not privacy-preserving. See Appendix B.3 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 users without their approval in any of her pictures. In this instantiation, since there are only two operational semantic rules, it is easy to discuss a possible modification in the rules so that $FP1$ is supported.

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

Tag - FR1

$$\text{FR1.1}\;\frac{\begin{gathered}\boldsymbol{FP1(tge) \notin \pi_{tge}} \qquad picture(o,\eta) \in KB(tgr) \\ KB'(o) = KB(o) \cup \{C^3_{\{o,tgr\}} tagRequest(tgr,tge,o,\eta)\} \\ KB'(tgr) = KB(tgr) \cup \{C^3_{\{o,tgr\}} tagRequest(tgr,tge,o,\eta)\}\end{gathered}}{SN \xrightarrow{tag(tgr,tge,picture(o,\eta))} SN'}$$

$$\text{FR1.2}\;\frac{\begin{gathered}\boldsymbol{FP1(tge) \in \pi_{tge}} \qquad picture(o,\eta) \in KB(tgr) \\ \boldsymbol{KB'(tge) = KB(tge) \cup \{C^3_{\{tge,tgr\}} tagRequest(tgr,tge,o,\eta)\}} \\ KB'(tgr) = KB(tgr) \cup \{C^3_{\{tge,tgr\}} tagRequest(tgr,tge,o,\eta)\}\end{gathered}}{SN \xrightarrow{tag(tgr,tge,picture(o,\eta))} SN'}$$

Accept tag request - FR2

$$\text{FR2.1}\;\frac{\begin{gathered}Au = audience(picture(o,\eta)) \cup friends(tge) \\ \boldsymbol{FP1(tge) \notin \pi_{tge}} \qquad a = removeTag_{tag(tge,tgr,o,\eta)} \\ acptr == o \qquad tagRequest(tge,tgr,o,\eta) \in KB(acptr) \\ A'_a = A_a \cup \{(o,o),(o,tge)\} \\ \forall i \in Au\; KB'(i) = KB(i) \cup \{C^3_{Au} tag(tge,tgr,o,\eta)\}\end{gathered}}{SN \xrightarrow{acceptTagRequest(acptr,tge,tgr,picture(o,\eta))} SN'}$$

$$\text{FR2.2}\;\frac{\begin{gathered}Au = audience(picture(o,\eta)) \cup friends(tge) \\ \boldsymbol{FP1(tge) \in \pi_{tge}} \qquad a = removeTag_{tag(tge,tgr,o,\eta)} \\ \boldsymbol{acptr == tge} \qquad tagRequest(tge,tgr,o,\eta) \in KB(acptr) \\ A'_a = A_a \cup \{(o,o),(o,tge)\} \\ \forall i \in Au\; KB'(i) = KB(i) \cup \{C^3_{Au} tag(tge,tgr,o,\eta)\}\end{gathered}}{SN \xrightarrow{acceptTagRequest(acptr,tge,tgr,picture(o,\eta))} SN'}$$

Table 4.2: New Tagging rules supporting FP1

to the taggee instead of the owner of the picture. Therefore

$$KB'(o) = KB(o) \cup \{C^3_{\{o,tgr\}} tagRequest(tgr, tge, o, \eta)\}$$

is replaced with

$$KB'(tge) = KB(tge) \cup \{C^3_{\{tge,tgr\}} tagRequest(tgr, tge, o, \eta)\}.$$

The resulting operational semantics rules are presented in table 4.2.

Finally if we include the new two rules in $\mathcal{FPPF}^{\mathcal{D}}_{\text{Facebook}}$ and we consider 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 FPU, which is the only policy available in the instantiation $\mathcal{FPPF}^{\mathcal{D}}_{\text{Facebook}}$. The only rule that can violate FPU 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 FPU cannot be violated. We refer the reader to Appendix B.4 for the complete proof. □

Summary of the Chapter

In this chapter we have introduced a formal notion of privacy-preserving SNS for $\mathcal{FPPF}$. Moreover, we have analysed the privacy of Twitter and Facebook by proving whether they preserve privacy under the presence of certain policies. We have also shown that the proofs give us useful information that can be used to update the instantiations we had for the SNSs, so that they support the new privacy policies.

As we have mentioned in previous chapters, in $\mathcal{FPPF}$ we do not use the traditional epistemic logic semantics. However, there exist similarities between classical epistemic logic and $\mathcal{FPPF}$. In the next chapter, we describe the semantics of first-order epistemic logic and we show some of the similarities of this logic to our approach.

Chapter 5

Relation of $\mathcal{FPPF}$ to Epistemic Logic

The way knowledge is interpredted in $\mathcal{FPPF}$ is not exactly as in classical epistemic logic. In this chapter, we first describe the conventional definition of first-order epistemic logic and later we discuss how it relates with $\mathcal{FPPF}$, in particular, social network models.

5.1 Epistemic Logic

Here we introduce background on First-Order Epistemic Logic (FOEL) very much as defined in [15]. As we showed in Chapter 2, the syntax of FOEL is that of First-Order Logic, but extended with the knowledge modality K_i. Let $\mathcal{T}$ be a set of functions symbols, predicate symbols and constants symbols. Function and predicate symbols have some implicit arity. Constant symbols can be seen as functions of arity 0. We refer to $\mathcal{T}$ as the vocabulary. Assume also an infinite supply of variables $x_1, \ldots, x_n$ ($n > 0$). Given a constant symbol c, function symbol f_i and variable x we define terms as $t ::= c \mid x \mid f_i(\overrightarrow{t})$.

Given the above and a set of agents AGT the language of FOEL, denoted as $\mathcal{L}_n$, is inductively defined as follows:

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

where $i \in AGT$.

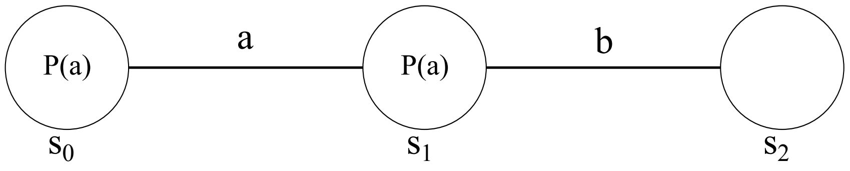

Figure 5.1: Relational Kripke structure

The semantics of FOEL is given by means of *relational Kripke structures*. However, in order to define relational Kripke structures we first need to introduce relational structures. A *relational structure* $\mathcal{T}$-structure, $\mathcal{A}$, consists of a nonempty set, denoted as $dom(\mathcal{A})$, called the domain, assignments of a k-ary relations $P^{\mathcal{A}} \subseteq dom(\mathcal{A})^k$ to the relation symbols in $\mathcal{T}$, an assignment of a k-ary function $f_i^{\mathcal{A}} : dom(\mathcal{A})^k \to dom(\mathcal{A})$ to each function symbol f_i of $\mathcal{T}$ and an assignment of a member $c^{\mathcal{A}}$ of the domain to each constant symbol c. Finally, we define a valuation function V on a structure $\mathcal{A}$, denoted $V^{\mathcal{A}}$, which maps each variable to an element in $dom(\mathcal{A})$, $V^{\mathcal{A}}(c) = c^{\mathcal{A}}$ for each c in $\mathcal{A}$ and we extend the definition by induction on the structure of terms $V^{\mathcal{A}}(f(t_1, \ldots, t_k)) = f^{\mathcal{A}}(V^{\mathcal{A}}(t_1), \ldots, V^{\mathcal{A}}(t_n))$.

Given the above, relational Kripke structures will consist of a set of *states* S, binary relations on them, denoted by $\mathcal{K}_i$, and a function π, which associates a $\mathcal{T}$-structure to each state $s \in S$. Formally, a relational Kripke structure for a vocabulary $\mathcal{T}$ and n agents is a tuple of the form $\langle S, \pi, \mathcal{K}_1, \ldots, \mathcal{K}_n \rangle$.

Example 8. *Let us consider a Kripke structure consisting of the agents a and b, the states s_0, s_1 and s_2, the predicate P with arity 1 and the relations $\mathcal{K}_a = \{(s_0, s_1), (s_1, s_0)\}$ and $\mathcal{K}_b = \{(s_1, s_2), (s_1, s_2)\}$. For the purpose of this example we assume that all relational structures $\pi(s_n)$ have a common domain $dom(\mathcal{A}) = \{a, b\}$. Moreover, $a \in P^{\pi(s_0)}$ and $a \in P^{\pi(s_1)}$. Fig. 5.1 shows a graphical representation of the described model.*

The semantics of fomulae written in $\mathcal{L}_n$, are defined as follows:

Definition 17. *Given a non-empty set of agents AGT, a Kripke structure $M = \langle S, \pi, \{\mathcal{K}_i\}_{i \in AGT} \rangle$, the agents $i \in AGT$, a finite set of agents $G \subseteq AGT$ and the valuation V, we define what it means for $\varphi \in \mathcal{L}_n$ under valuation V to be true, written $(M, s, V) \models \varphi$, as shown in Table 5.1.* □

$(M, s, V) \models P(t_1, \ldots, t_k)$	iff	$(V^{\pi^c(s)}(t_1), \ldots, V^{\pi^c(s)}(t_k)) \in P^{\pi^c(s)}$
$(M, s, V) \models \neg\varphi$	iff	$(M, s, V) \not\models \varphi$
$(M, s, V) \models \varphi_1 \wedge \varphi_2$	iff	$(M, s, V) \models \varphi_1$ and $(M, s, V) \models \varphi_2$
$(M, s, V) \models \forall x.\varphi$	iff	for all $v \in dom(\mathcal{A})$, $(M, s, V) \models \varphi[v/x]$
$(M, s, V) \models K_i\varphi$	iff	$(M, t, V) \models \varphi$ for all t such that $(s, t) \in \mathcal{K}_i$

Table 5.1: $\mathcal{KBL}_{\mathcal{SN}}$ satisfiability relation over Kripke structures

As we showed in Table 5.1, in order for agents to *know* some formula φ, it must be true in all states that are accessible for her. Intuitively, relational Kripke structures model the *uncertainty* of the agents.

Example 9. *Let M be the model presented in Fig. 5.1. It holds that $(M, s_0, V) \models K_aP(a)$. Indeed, since $P(a)$ holds in all states accessible for a from s_0, which in M, is only s_0. Therefore $(M, s_0, V) \models P(a)$.*

It also holds that $(M, s_0, V) \models \neg K_bP(a)$, since in one of the states that b considers possible $P(a)$ is not true. In particular, $(M, s_2, V) \models \neg P(a)$.

This semantics for knowledge has been given several axiomatisations. These axiomatisations are based on *properties of knowledge* (or axioms) that hold depending on the type of binary relations in the Kripke structure. For instance, if we define the binary relation to be reflexive the following axiom holds.

$$K_i\varphi \implies \varphi$$

A list of the most common axiomatisations for knowledge and their proofs of soundness and completeness with respect to their respective Kripke structures can be found in [15].

5.2 $\mathcal{FPPF}$ vs Epistemic Logic

The most notorious difference between the semantics of $\mathcal{FPPF}$ and FOEL is the way knowledge is interpreted. SNMs "store" in each node the $\mathcal{F}_{\mathcal{KBL}}$ formulae that the agents know. On the contrary, in relational Kripke structures, the *uncertainty* of the agents is modelled by means of the binary relation $\mathcal{K}_i$.

Nevertheless, it does not mean that the two models are complementary. In [15] Fagin *et al.* show how to construct *knowledge* *bases* for systems consisting of several agents by using what they called *knowledge-based programming*. They defined the state of an agent to be a tuple which contains all formulae the agent knows at a particular moment in time. The SNMs defined here, contain the same information, but also additional information regarding permissions and connections between users. As a matter of fact, we claim 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 [15] resulting from φ can be built.

In the case of individual agents this fact becomes clearer. In each agent node there are several formulae explicitly representing what she knows. As we mentioned in Section 2.1.2, all predicates are interpreted equally when they are placed inside the knowledge base of the agent. Then we provide the agent with a deductive engine, which allows her to derive new knowledge from the explicit formulae. In particular, we assume that agents can infer new knowledge using any of the derivation rules or axioms in **K**. **K** corresponds to the set of Kripke structures where $\mathcal{K}$ does not have any restrictions. If we had chosen the axiomatisation **T**, we would have needed the set of relational Kripke structures where $\mathcal{K}$ is reflexive. **T** is composed by the same set of derivation rules and axioms than **K**, but it also includes the axiom $K_i\varphi \implies \varphi$. Hence each agent node can easily be encoded in a relational Kripke structure with the appropriate restrictions in their binary relations, which depend on the properties of knowledge we assume in the deductive engine.

In Section 2.1.2 we wrote that we cannot just assume the axiomatisation **K** for $\mathcal{KBL}_{\mathcal{SN}}$ in general (i.e. not only of knowledge of individual agents, but also for global knowledge in the SNM). The reason is because there are properties of knowledge that hold in relational Kripke structures that do not hold in SNMs. Consider the necessitation rule from **K**

$$\frac{\varphi}{K_i\varphi} \tag{5.1}$$

where i is any agent in the system. The meaning of the rule is that given a formula φ, which is valid (i.e. φ holds at any state of a relational Kripke structure), then $K_i\varphi$ is also valid. Intuitively, all agents in the system know all tautologies (i.e. formulae which are always true). This rule holds even in the simplest relational Kripke structure, which is the one with no restrictions in its binary relations.

Consider now a relational Kripke structure where the predicate $friends(Alice, Bob)$ is valid. According to the rule (5.1), the formula $K_i friends(Alice, Bob)$ is also valid for all $i \in AGT$. On the other hand, let us define validity of a formula φ in SNMs in a similar way, i.e. φ is valid if it is true at any node in the SNM; formally, $\forall u \in Ag\ SN, u \models \varphi$. In this case the rule (5.1) would not hold. In order for the predicate $friends(Alice, Bob)$ to be true at all nodes, it is only needed that $(Alice, Bob) \in C_{Friends}$. It could be possible that $(Alice, Bob) \in C_{Friends}$ holds but $\neg K_{Bob} friends(Alice, Bob)$, since the two facts are unrelated in SNMs. Therefore we can conclude that (5.1) is not sound with respect to SNMs.

These examples show why we cannot simply assume that axiomatisations that are sound and complete for relational Kripke structures can directly be used in $\mathcal{KBL}_{\mathcal{SN}}$ and SNMs globally, but they hold for the internal knowledge of individual agents.

Summary of the Chapter

In this chapter we have provided the basics of FOEL and we have described the similarities and differences with our SNMs. Moreover, we have "informally" argued why we cannot directly use axiomatisations from FOEL directly in $\mathcal{KBL}_{\mathcal{SN}}$. In the next chapter, we present related work regarding the use of epistemic logic to model knowledge SNSs (but not specific for privacy) and an access control mechanism for SNSs.

Chapter 6

Concluding Discussion

In this chapter we discuss related work about the use of epistemic logic to model knowledge in SNSs and the current protection mechanism in SNSs. Finally, we conclude with the conclusions and the future work.

6.1 Related Work

In this section we describe applications of epistemic logic in security and other approaches to model SNSs using dynamic epistemic logic. We also discuss a formalism developed by Fong. *et al.* which describes the current access control mechanism present in most SNSs nowadays.

6.1.1 Epistemic Logic and SNSs

Epistemic logic has been widely used for analysing security and privacy properties in multi-agent systems (MAS) in the past. Traditionally the evolution of knowledge in epistemic logic is modelled by means of run and events, in a "run-and-systems" framework, known as *Interpreted systems* [15]. In [19], Halpern *et al.* use Interpreted Systems to formalise the notion of secrecy in MAS. In their work 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 [9], where Balliu presents a knowledge-based account to specify information flow conditions in a distributed setting. The main advantage of his 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 possible to verify real systems. For instance, MCK [18] and MCMAS [23] are state of the art model checkers for temporal-epistemic logics based on Interpreted systems. They have successfully been used to verify security properties in several cryptographic protocols. However, we are not aware of any specific use for verifying privacy policies.

Interpreted systems are able to represent the knowledge at different points in time. But there are no formal definitions of the events that can be executed in order to specify how knowledge evolves. Instead they require a description of the protocol which describes the evolution of knowledge. *Dynamic Epistemic Logic* (DEL) provides a basis for operations of knowledge evolution in epistemic logic [10, 29]. 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 [32] 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 [31], DEL is used with the intended aim of studying 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 of revolt or stay at home.

DEL turns out to be not well-suited in our setting. Firstly, because it is based in the classical Kripke semantics for epistemic logic [15]. As we have described in the previous chapter, 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 SN dependent factors. As described in Section 3.2.1, we use the premises of the rules when stating the conditions for each event.

6.1.2 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, which aimes at providing a better understating of ReBAC [17, 16]. In their work, Fong *et al.* developed 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 where the type of the relationship is also taken into account (e.g. patient-physician, parent-child). In addition, they also 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 the relationships between users depending on the access context. The audience of the resources is defined by means of ReBAC policies. In [11] 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 before. 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.

6.2 Conclusions and Future Work

In this thesis 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 evolving social networks, by means of a labelled transition system. The framework was enhanced with a set of operational semantics rules, which were instantiated for Twitter. We have shown how a designer can use our dynamic framework to model evolving 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, which was done by analysing the information of the earlier proof where we showed that Facebook did 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, a designer could write a dynamic instantiation of the SNS that she wants to implement. Then, she 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 additional enforcement is required.

On the other hand, we would like to provide a run-time enforcement mechanism for SNSs under consideration. 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 would check 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* [1, 5] to show the practicality of our approach.

Relation to Standard Semantics for Epistemic Logic

As we have shown in Chapter 5, there are a lot of open questions when it comes to the relation of SNMs and relational Kripke structures. We are currently investigating this questions. In particular, the two main problems we are addressing are:

i) Whether there is a way to encode our SNMs into relational Kripke structures.

ii) Specify a sound and complete axiomatisation (similar to the ones for traditional epistemic logic) of $\mathcal{KBL}_{\mathcal{SN}}$ with respect to SNMs.

There are other issues that we do not mention in this thesis, but we plan to study in the future. For instance, the properties of common and distributed knowledge in SNMs or developing and studying the complexity of a model checking algorithm.

Privacy Policies and Time

Privacy policies in $\mathcal{FPPF}$ cannot express real time properties. For example, a user may want to write a policy which says "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 this 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. We should 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 we should be able to tell apart these two locations in $\mathcal{FPPF}$. It would also solve other issues present in our current notion of privacy-preserving SNS. It is too strong in some cases. For instance, imagine that Alice shared her location with Bob and Charlie a month ago. Now Alice is going on vacation but she does not want anybody to know it, therefore she decides to activate the policy "nobody can know my location". Keeping the privacy-preserving property in $\mathcal{FPPF}$ would require forbidding Alice to activate the privacy policy, since her location is already known by Bob and Charlie and the resulting SNM would not be in conformance with the policy. In order to solve this problem, we should not only be able to differentiate between the old and new location, but also we should know when the policy was activated. We are actively working on this extension for $\mathcal{FPPF}$.

Bibliography

[1] Diaspora*. https://joindiaspora.com/. Accessed: 2015-04-12.

[2] Directive 95/46/ec of the european parliament and of the council of 24 october 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data. http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:31995L0046:en:HTML. Accessed: 2015-07-01.

[3] Facebook urged to tighten privacy settings after harvest of user data, the guardian. http://www.theguardian.com/technology/2015/aug/09/facebook-privacy-settings-users-mobile-phone-number. Accessed: 2015-09-05.

[4] Here's how to use facebook's mystifying privacy settings, wired. http://www.wired.com/2015/08/how-to-use-facebook-privacy-settings-step-by-step/?mbid=social_fb. Accessed: 2015-09-05.

[5] $\mathcal{PPF}$ Diaspora*. https://github.com/raulpardo/ppf-diaspora. 2015.

[6] The universal declaration of human rights. http://www.un.org/en/documents/udhr/. Accessed: 2015-09-04.

[7] What you need to know about instagram privacy settings, thestar.com. http://www.thestar.com/news/privacy-blog/2015/02/what-you-need-to-know-about-instagram-privacy-settings.html. Accessed: 2015-09-05.

[8] Irwin Altman. Privacy: A conceptual analysis. *Environment and behavior*, 8(1):7–29, 1976.

[9] Musard Balliu. A logic for information flow analysis of distributed programs. In *Secure IT Systems*, pages 84–99. Springer, 2013.

[10] Johan van Benthem, Jan van Eijck, and Barteld Kooi. Logics of communication and change. *Information and computation*, 204(11):1620–1662, 2006.

[11] Glenn Bruns, Philip WL Fong, Ida Siahaan, and Michael Huth. Relationship-based access control: its expression and enforcement through hybrid logic. In *CODASPY'12*, pages 117–124. ACM, 2012.

[12] danah boyd and Nicole B. Ellison. Social network sites: Definition, history, and scholarship. *Journal of Computer-Mediated Communication*, 13:210–230, 2008.

[13] Bernhard Debatin. Ethics, privacy, and self-restraint in social networking. In Sabine Trepte and Leonard Reinecke, editors, *Privacy Online*, pages 47–60. Springer Berlin Heidelberg, 2011.

[14] Kayhan Erciyes. *Complex Networks: An Algorithmic Perspective*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 2014.

[15] Ronald Fagin, Joseph Y Halpern, Yoram Moses, and Moshe Y Vardi. *Reasoning about knowledge*, volume 4. MIT press Cambridge, 2003.

[16] Philip W.L. Fong. Relationship-based access control: Protection model and policy language. In *CODASPY'11*, pages 191–202. ACM, 2011.

[17] PhilipW.L. Fong, Mohd Anwar, and Zhen Zhao. A privacy preservation model for facebook-style social network systems. In *ESORICS'09*, LNCS, pages 303–320. Springer, 2009.

[18] Peter Gammie and Ron van der Meyden. Mck: Model checking the logic of knowledge. In Rajeev Alur and DoronA. Peled, editors, *Computer Aided Verification*, volume 3114 of *Lecture Notes in Computer Science*, pages 479–483. Springer Berlin Heidelberg, 2004.

[19] Joseph Y Halpern and Kevin R O'Neill. Secrecy in multiagent systems. *ACM Transactions on Information and System Security (TISSEC)*, 12(1):5, 2008.

[20] Maritza Johnson, Serge Egelman, and Steven M. Bellovin. Facebook and privacy: It's complicated. SOUPS '12, pages 9:1–9:15, New York, NY, USA, 2012. ACM.

[21] Amanda Lenhart, Kristen Purcell, Aaron Smith, and Kathryn Zickuhr. Social media & mobile internet use among teens and young adults. *Pew Internet & American Life Project*, 2010.

[22] Yabing Liu, Krishna P. Gummadi, Balachander Krishnamurthy, and Alan Mislove. Analyzing facebook privacy settings: User expectations vs. reality. IMC '11, pages 61–70. ACM, 2011.

[23] Alessio Lomuscio, Hongyang Qu, and Franco Raimondi. Mcmas: A model checker for the verification of multi-agent systems. In *Computer Aided Verification*, pages 682–688. Springer, 2009.

[24] M. Madejski, M. Johnson, and S.M. Bellovin. A study of privacy settings errors in an online social network. (PERCOM Workshops'12), pages 340–345.

[25] Michelle Madejski, Maritza Lupe Johnson, and Steven Michael Bellovin. The failure of online social network privacy settings. 2011.

[26] John-Jules Ch Meyer and Wiebe Van Der Hoek. *Epistemic Logic for AI and Computer Science*. Cambridge University Press, New York, NY, USA, 1995.

[27] Raul Pardo, Musard Balliu, and Gerardo Schneider. A formal approach to preserving privacy in social networks (extended version). Technical report, Chalmers University of Technology.

[28] Raúl Pardo and Gerardo Schneider. A formal privacy policy framework for social networks. In *SEFM'14*, volume 8702 of *LNCS*, pages 378–392. Springer, 2014.

[29] Jan Plaza. Logics of public communications. *Synthese*, 158(2):165–179, 2007.

[30] Riccardo Pucella. Knowledge and security. *arXiv preprint arXiv:1305.0876*, 2013.

[31] Ji Ruan and Michael Thielscher. A logic for knowledge flow in social networks. In *AI 2011: Advances in Artificial Intelligence*, pages 511–520. Springer, 2011.

[32] Jeremy Seligman, Fenrong Liu, and Patrick Girard. Facebook and the epistemic logic of friendship. In *TARK'13*, 2013.

[33] Daniel J Solove. Understanding privacy. 2008.

[34] Samuel D Warren and Louis D Brandeis. The right to privacy. *Harvard law review*, pages 193–220, 1890.

[35] Alan F Westin. Privacy and freedom. *Washington and Lee Law Review*, 25(1):166, 1968.

Appendices

Appendix A

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.

A.1 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 users can block other users. This pair represents the events of blocking and unblocking a user, respectively.

- $showReco$ - 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 sematnics rules for each of the events in EVT_{Twitter}.

A.2 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.

A.2.1 Epistemic

R1 - Tweet

R1.1.1

$$\frac{\begin{array}{c} Au = followers(tu) \cup \{u\} \cup \{u \mid mention(u, tu, \eta) \in TweetInfo\} \\ state(tu) == \text{'Public'} \qquad Inclocation(u) == true \\ \forall p(\vec{t}) \in TweetInfo\ \forall i \in Au\ KB'(i) = KB(i) \cup \{C^3_{Au} p(\vec{t})\} \end{array}}{\langle \_, \_, KB, \_\rangle \xrightarrow{tweet(tu, TweetInfo)} \langle \_, \_, KB', \_\rangle}$$

R1.2.2

$$\frac{\begin{array}{c} Au = followers(tu) \cup \{u\} \qquad state(tu) == \text{'Private'} \\ Inclocation(u) == false \qquad location(tu, \eta) \notin TweetInfo \\ \forall p(\vec{t}) \in TweetInfo\ \forall i \in Au\ KB'(i) = KB(i) \cup \{C^3_{Au} p(\vec{t})\} \end{array}}{\langle \_, \_, KB, \_\rangle \xrightarrow{tweet(tu, TweetInfo)} \langle \_, \_, KB', \_\rangle}$$

R1.1.2

$$\frac{\begin{array}{c} Au = followers(tu) \cup \{u\} \cup \{u \mid mention(u, tu, \eta) \in TweetInfo\} \\ state(tu) == \text{'Public'} \\ Inclocation(u) == false \qquad location(tu, \eta) \notin TweetInfo \\ \forall p(\vec{t}) \in TweetInfo\ \forall i \in Au\ KB'(i) = KB(i) \cup \{C^3_{Au} p(\vec{t})\} \end{array}}{\langle \_, \_, KB, \_\rangle \xrightarrow{tweet(tu, TweetInfo)} \langle \_, \_, KB', \_\rangle}$$

R1.2.1

$$\frac{\begin{array}{c} Au = followers(tu) \cup \{u\} \\ state(tu) == \text{'Private'} \qquad Inclocation(u) == true \\ \forall p(\vec{t}) \in TweetInfo\ \forall i \in Au\ KB'(i) = KB(i) \cup \{C^3_{Au} p(\vec{t})\} \end{array}}{\langle \_, \_, KB, \_\rangle \xrightarrow{tweet(tu, TweetInfo)} \langle \_, \_, KB', \_\rangle}$$

R2 - Retweet

R2.1

$$\frac{\begin{array}{c} F = getTweetInfo(tu, \eta) \\ state(tu) == \text{'Public'} \qquad state(rtu) == \text{'Public'} \\ TweetInfoAu = followers(tu) \cup followers(rtu) \cup \{tu, rtu\} \\ RetweetAu = followers(tu) \cup followers(rtu) \cup \{tu, rtu\} \\ tweet(tu.\eta) \in KB(rtu) \\ \forall p(\vec{t}) \in F\ \forall i \in TweetInfoAu\ KB'(i) = KB(i) \cup \{C^3_{Au} p(\vec{t})\} \\ \forall i \in RetweetAu\ KB'(i) = KB(i) \cup \{C^3_{RetweetAu} retweet(rtw, tu, \eta)\} \end{array}}{\langle \_, \_, KB, \_\rangle \xrightarrow{retweet(rtu, tweet(tu, \eta))} \langle \_, \_, KB', \_\rangle}$$

R2.2

$$\frac{\begin{array}{c} F = getTweetInfo(tu,\eta) \\ state(tu) == \text{'Public'} \qquad state(rtu) == \text{'Private'} \\ TweetInfoAu = followers(tu) \cup followers(rtu) \cup \{tu, rtu\} \\ RetweetAu = followers(rtu) \cup \{rtu\} \\ tweet(tu.\eta) \in KB(rtu) \\ \forall p(\vec{t}) \in F\ \forall i \in TweetInfoAu\ KB'(i) = KB(i) \cup \{C^3_{Au} p(\vec{t})\} \\ \forall i \in RetweetAu\ KB'(i) = KB(i) \cup \{C^3_{RetweetAu} retweet(rtw, tu, \eta)\} \end{array}}{\langle \_, \_, KB, \_ \rangle \xrightarrow{retweet(rtu, tweet(tu,\eta))} \langle \_, \_, KB', \_ \rangle}$$

R3 - Favourite

R3

$$\frac{\begin{array}{c} tweet(tu,\eta) \in KB(fu) \\ \forall i \in \{fu, tu\}\ KB'(i) = KB(i) \cup \{favourite(fu, tu, \eta)\} \end{array}}{\langle \_, \_, KB, \_ \rangle \xrightarrow{favourite(fu, tweet(tu,\eta))} \langle \_, \_, KB', \_ \rangle}$$

R4 - Access profile

R4.1

$$\frac{\begin{array}{c} F = Info(acd) \\ [(acr, acd) \in A_{accessProf} \vee (acr, acd) \in A_{accessProfRec}] \\ Status(acd) = \text{'Public'} \\ \forall p(\vec{t}) \in F\ KB'(acr) = KB(acr) \cup \{p(\vec{t})\} \end{array}}{\langle \_, \{\{A_i\}_{i \in \Sigma}, \_\}, KB, \_ \rangle \xrightarrow{accessProf(acr, acd)} \langle \_, \{\{A_i\}_{i \in \Sigma}, \_\}, KB', \_ \rangle}$$

R4.2

$$\frac{\begin{array}{c} F = Info(acd) \\ [(acr, acd) \in A_{accessProf} \vee (acr, acd) \in A_{accessProfRec}] \\ Status(acd) = \text{'Private'} \\ (acd, acr) \in C_{Follower} \qquad \forall p(\vec{t}) \in F\ SN', acr \models K_{acr} p(\vec{t}) \end{array}}{\langle \_, \{\{A_i\}_{i \in \Sigma}, \{\{C_i\}_{i \in \mathcal{C}}, \_\}, KB, \_ \rangle \xrightarrow{accessProf(acr, acd)} \langle \_, \{\{A_i\}_{i \in \Sigma}, \{C_i\}_{i \in \mathcal{C}}, \_\} KB', \_ \rangle}$$

R10 - Show recommendation

R10.1

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

R10.2

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

R11 - Show advertisment

R11

$$\frac{\begin{array}{c} (advertiser, user) \in A_{sendAd} \\ KB'(user) = KB(user) \cup \{advertise(advertiser, \eta)\} \end{array}}{\langle \_, \{\{A_i\}_{i \in \Sigma}, \_\}, KB, \_\rangle \xrightarrow{showAdv(advertiser, user)} \langle \_, \{\{A_i\}_{i \in \Sigma}, \_\}, KB', \_\rangle}$$

A.2.2 Topological

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

R6.1

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

R7 - Accept follow request

R7

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

R8 - Block

R8.1

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

R8.2

$$\frac{(blocker, blocked) \in C_{Follower} \qquad (blocker, blocked) \notin C_{Block} \\ C'_{Block} = C_{Block} \cup \{(blocker, blocked)\} \\ C'_{Followers} = C_{Followers} \setminus \{(blocker, blocked)\}}{\langle \_, \{\{C_i\}_{i\in\mathcal{C}}, \_\}, \_, \_\rangle \xrightarrow{block(blocker,blocked)} \langle \_, \{\{C'_i\}_{i\in\mathcal{C}}, \_\}, \_, \_\rangle}$$

R9 - Unblock

R9

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

A.2.3 Policy

R14 - Change state to private

R14

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

R15 - Change state to public

R15

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

R16 - Include location on Tweets

R16

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

R17 - Not include location on Tweets

R17

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

A.2.4 Hybrid

R5 - Create profile

R5

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

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

R6.2

$$\frac{\begin{array}{c} (followed, follower) \notin C_{Block} \qquad state(followed) == \text{'Private'} \\ (follower, followed) \notin C_{Follower} \\ \forall i \in \{followed, follower\}\ KB'(i) = KB(i) \cup \{C^3_{\{followed, follower\}} followRequest(follower)\} \end{array}}{\langle \_, \{\{C_i\}_{i \in \mathcal{C}}, \_\}, KB, \_ \rangle \xrightarrow{follow(follower, followed)} \langle \_, \{\{C_i\}_{i \in \mathcal{C}}, \_\}, KB', \_ \rangle}$$

R12 - Allow advertisment

R12

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

R13 - Disallow advertisement

R13

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

Appendix B

Proofs

B.1 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. Proposition to be disclosed $TweetInfo \subseteq 2^{\mathcal{P}}$ where $tweet(u, \eta) \in TweetInfo$

9.1.3. $e = tweet(u, TweetInfo)$

9.1.4. We want to prove:

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

9.2. By contradiction, let us assume

9.2.1. $SN \models_C P1(u)$ and $SN \xrightarrow{e} SN'$

9.2.2. $SN' \not\models_C P1(u)$

9.3. By 18.2.2.

9.3.1. $SN' \not\models_C P1(u)$ [Def. $\models_C$]

9.3.2. $SN', u \models \neg\neg S_{Ag \setminus followers(u) \setminus \{u\}} tweet(u, \eta)$ $[\neg\neg_e]$

9.3.3. $SN', u \models S_{Ag \setminus followers(u) \setminus \{u\}} tweet(u, \eta)$

9.4. By 18.3.5. and the definition of $\models$ we have

9.4.1. $\exists i \in Ag \setminus followers(u) \setminus \{u\}$ s.t. $SN', i \models K_i tweet(u, \eta)$

9.5. By Def. of $R1$, we have that

9.5.1. $\forall p(\overrightarrow{t}) \in TweetInfo\ SN', u \models C^3_{followers(u) \cup \{u\}} p(\overrightarrow{t})$ [By 9.1.2.]

9.5.2. $SN', u \models C^3_{followers(u) \cup \{u\}} tweet(u, \eta)$ [By $\models$]

9.5.3. $SN', u \models E^0_{followers(u) \cup \{u\}} tweet(u, \eta) \wedge$
$E^1_{followers(u) \cup \{u\}} tweet(u, \eta) \wedge$
$E^2_{followers(u) \cup \{u\}} tweet(u, \eta) \wedge$
$E^3_{followers(u) \cup \{u\}} tweet(u, \eta)$ [By $\models$]

9.5.4. $SN', u \models E^1_{followers(u) \cup \{u\}} tweet(u, \eta)$ [By $\models$]

9.5.5. $\forall j \in followers(u) \cup \{u\}\ SN, j \models K_j tweet(u, \eta)$

9.6. By 18.2.1. we have

9.6.1. $SN \models_C P1(u)$ [By $\models_C$]

9.6.2. $SN, u \models \neg S_{Ag \setminus followers(u) \setminus \{u\}} tweet(u, \eta)$ [By Def. S_G]

9.6.3. $SN, u \models \neg(\bigvee_{i \in Ag \setminus followers(u) \setminus \{u\}} K_i tweet(u, \eta))$ [Morgan]

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

9.7. By 9.6.4. and 18.5.4. we have

9.7.1. $SN' \models_C P1(u)$

9.8. By 18.2.2. and 9.7.1. we derive a contradiction. □

10. <u>Executing $R1$ and $P3$ enabled</u>

10.1. Given

10.1.1. $u \in Ag$ (owner of the privacy policy $P3(u)$)

10.1.2. Propositions to be disclosed $TweetInfo \subseteq 2^{\mathcal{P}}$

10.1.3. Location of the tweet $location(u, \eta)$

10.1.4. $Au \subseteq Ag$

10.1.5. $e = tweet(u, TweetInfo)$

10.1.6. We want to prove:

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

10.2. By contradiction, let us assume

10.2.1. $SN \models_C P3(u)$ and $SN \xrightarrow{e} SN'$

10.2.2. $SN' \not\models_C P3(u)$

10.3. By 10.2.1. and $\models_C$

10.3.1. $SN', u \models \neg\neg S_{Ag\setminus\{u\}} location(u, \eta)$ $[\neg\neg_e]$

10.3.2. $SN', u \models S_{Ag\setminus\{u\}} location(u, \eta)$ [By $\models$]

10.3.3. $\exists i \in Ag \setminus \{u\}$ such that $SN', i \models K_i location(u, \eta)$

10.4. By Def. of $R1$

10.4.1. $\forall p(\overrightarrow{t}) \in TweetInfo \setminus \{location(u, \eta)\}$ $SN', u \models C^3_{Au} p(\overrightarrow{t})$

10.5. By 10.2.1. and the definition of $\models_C$

10.5.1. $SN, u \models \neg S_{Ag\setminus\{u\}} location_\eta$ [Def. S_G]

10.5.2. $SN, u \models \neg(\bigvee_{i \in Ag\setminus\{u\}} K_i location(u, \eta))$ [Morgan]

10.5.3. $SN, u \models \bigwedge_{i \in Ag\setminus\{u\}} \neg K_i location(u, \eta)$

10.6. By 10.4.1. and 10.5.3.

10.6.1. $SN' \models_C P3(u)$

10.7. By 10.6.1. and 10.2.2. we derive a contradiction. □

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

11. Executing $R2$ and $P2$ enabled

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:

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

11.2. By contradiction, let us assume
11.2.1. $SN \models_C P2(u)$ and $SN \xrightarrow{e} SN'$
11.2.2. $SN' \not\models_C P2(u)$

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

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

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

11.6. By 11.4.5. and 11.5.3.

11.6.1. $SN' \models_C P2(u)$

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

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

12.2. By contradiction, let us assume
12.2.1. $SN \models_C P3(u)$ and $SN \xrightarrow{e} SN'$
12.2.2. $SN' \not\models_C P3(u)$

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

12.4. By Def. of $R2$
12.4.1. $\forall p(\overrightarrow{t}) \in getTweetInfo(tu, \eta) \setminus \{location(tu, \eta)\}$ $SN', rtu \models C^3_{TweetInfoAu} p(\overrightarrow{t})$

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

12.6. By 12.4.1. and 12.5.3.
12.6.1. $SN' \models_C P3(u)$

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

$R3$ – None of the privacy policies in Def. 12 can be violated by rule $R3$

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

$R4$ – The execution of $R4$ could violate the privacy policies $P1$, $P2$, $P3$

13. <u>Executing $R4$ and $P1$ enabled</u>

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:

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

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. $SN \models_C P1(acd)$ and $SN \xrightarrow{e} SN'$

13.3.2. $SN' \not\models_C P1(acd)$

13.4. By 13.3.2.

13.4.1. $SN' \not\models_C P1(acd)$ [Def. $\models_C$]

13.4.2. $SN', u \models \neg\neg S_{Ag \setminus followers(u) \setminus \{u\}} u.tweet_\eta$ [$\neg\neg_e$]

13.4.3. $SN', u \models S_{Ag \setminus followers(u) \setminus \{u\}} u.tweet_\eta$[By $\models$]

13.4.4. $\exists i \in Ag \setminus followers(u) \setminus \{u\}$ s.t. $SN', i \models K_i tweet(acd, \eta)$

13.5. By 13.3.1., 13.1.3., Def. of $state$ and Def. of $R4$, we have that

13.5.1. If $(acr, acd) \in C_{Follower}$

13.5.1.1. $\forall p(\overrightarrow{t}) \in Info(acd)\ SN', acr \models K_{acr}p(\overrightarrow{t})$ [By 13.2.1.]

13.5.1.2. $SN', acr \models K_{acr}tweet(acd, \eta)$

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. $SN, acd \models \neg S_{Ag \setminus followers(acd) \setminus \{acd\}} tweet(acd, \eta)$ [By Def. S_G]

13.6.3. $SN, acd \models \neg(\bigvee_{i \in Ag \setminus followers(acd) \setminus \{acd\}} K_i tweet(acd, \eta))$ [Morgan]
13.6.4. $SN, acd \models \bigwedge_{i \in Ag \setminus followers(acd) \setminus \{acd\}} \neg K_i tweet(acd, \eta)$

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 $acd.retweet(acd, u, \eta)$ 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 Def. 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:

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

15.2. By contradiction, let us assume
15.2.1. $SN \models_C P4(r)$ and $SN \xrightarrow{e} SN'$
15.2.2. $SN' \not\models_C P4(r)$

15.3. By 15.2.2.
15.3.1. $SN' \not\models_C P4(r)$ [By $\models_C$]
15.3.2. $SN', r \models \neg\forall x.(\neg K_x(\mathit{email}(r) \vee \mathit{phone}(r)) \implies \neg P_x^r \mathit{accessProfRec})$ [By $\neg\forall z.\varphi \equiv \exists z.\neg\varphi$]
15.3.3. $SN', r \models \exists x.\neg(\neg K_x(\mathit{email}(r) \vee \mathit{phone}(r)) \implies \neg P_x^r \mathit{accessProfRec})$ [By $\exists_e$, where $\varphi[v/x]$]
15.3.4. $SN', r \models \neg(\neg K_v(\mathit{email}(r) \vee \mathit{phone}(r)) \implies \neg P_v^r \mathit{accessProfRec})$ [By $\models$]
15.3.5. $SN', r \models \neg(\neg(\neg K_v(\mathit{email}(r) \vee \mathit{phone}(r))) \vee (\neg P_v^r \mathit{accessProfRec}))$ [$\neg\neg_e$]
15.3.6. $SN', r \models \neg(K_v(\mathit{email}(r) \vee \mathit{phone}(r)) \vee \neg P_v^r \mathit{accessProfRec})$ [Morgan]
15.3.7. $SN', r \models \neg K_v(\mathit{email}(r) \vee \mathit{phone}(r)) \wedge P_v^r \mathit{accessProfRec}$

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

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

15.6. By 15.5.6. and 15.4.1.1.

15.6.1. $SN' \models_C P4(r)$

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

$R11 - R13$ – None of the privacy policies in Def. 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 Def. 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. □

B.2 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 $SN \models_C P6(u)$, 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 $SN' \not\models_C P6(u)$.

Proof.

16. <u>Executing R1 and P6 activated</u>

16.1. Given
16.1.1. User $u \in Ag$ such that $SN \models_C P6(u)$
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

$$SN \models_C P6(u) \text{ and } SN \xrightarrow{e} SN' \text{ then } SN' \not\models_C P6(u)$$

16.2. Let us assume
16.2.1. $SN' \not\models_C P6(u)$ [By $\models_C$]
16.2.2. $SN', u \models \neg(K_i location(tu, \eta) \wedge K_i mention(u, tu, \eta))$

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

16.4. By the Def. of $R1$ and 17.1.7.
16.4.1. If $state(tu) ==$ *'Public'*
16.4.2. $\forall p(\vec{t}) \in TweetInfo\ SN', u \models C^3_{followers(tu)\cup\{tu\}} p(\vec{t})$ [By 16.1.3.]
16.4.3. $SN', u \models C^3_{followers(tu)\cup\{tu\}} location(tu, \eta) \wedge mention(u, tu, \eta)$ [By $\models$]
16.4.4. $\forall i \in followers(tu) \cup \{tu\} SN', u \models K_i location(tu, \eta) \wedge mention(u, tu, \eta)$

16.4.5. If $state(tu) ==$ *'Private'*
16.4.6. $\forall p(\vec{t}) \in TweetInfo\ SN', u \models C^3_{followers(tu)\cup\{tu\}\cup\{u\}} p(\vec{t})$ [By 16.1.3.]
16.4.7. $SN', u \models C^3_{followers(tu)\cup\{tu\}\cup\{u\}} location(tu, \eta) \wedge mention(u, tu, \eta)$ [By $\models$]
16.4.8. $\forall i \in followers(tu) \cup \{tu\} \cup \{u\} SN', u \models K_i location(tu, \eta) \wedge mention(u, tu, \eta)$

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

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

□

B.3 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 $\mathcal{FPPF}^{\mathcal{D}}_{\text{Facebook}}$) 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 $\mathcal{FPPF}^{\mathcal{D}}_{\text{Facebook}}$ 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 $SN, SN' \in \mathcal{SN}_{\text{Facebook}}$ 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 $SN', o \models E_{Au} tag(tge, o, o, \eta)$, hence $SN' \not\models_C FP1(tge)$, which contradicts our claim $SN' \models_C FP1(tge)$ and therefore $\mathcal{FPPF}^{\mathcal{D}}_{\text{Facebook}}$ 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. $SN', tge \models \forall o.\forall t.\forall \eta.(\neg K_{tge} tagRequest(t, tge, o, \eta) \implies \neg S_{Ag} tag(tge, t, o, \eta))$ [By Implication equivalence]

17.2.3. $SN', tge \models \forall o.\forall t.\forall \eta.(K_{tge} tagRequest(t, tge, o, \eta) \vee \neg S_{Ag} tag(tge, t, o, \eta))$ [By $\neg\neg_i$]

17.2.4. $SN', tge \models \forall o.\forall t.\forall \eta.\neg\neg(K_{tge} tagRequest(t, tge, o, \eta) \vee \neg S_{Ag} tag(tge, t, o, \eta))$ [By Morgan]

17.2.5. $SN', tge \models \forall o.\forall t.\forall \eta.\neg(\neg K_{tge} tagRequest(t, tge, o, \eta) \wedge S_{Ag} tag(tge, t, o, \eta))$

17.3. Let us assume

17.3.1. $SN \models_C FP1(tge)$ and $SN \xrightarrow{e} SN'$

17.4. By the Def. of $FR1$, 17.1.2.

17.4.1. $SN', tge \models \neg K_{tge} tagRequest(tgr, tge, o, \eta)$

17.5. By the Def. of $FR1$, 17.1.7.

17.5.1. $SN', o \models C^3_{Au} tag(tge, tgr, o, \eta)$[By $\models$]

17.5.2. $SN', o \models E^3_{Au} tag(tge, tgr, o, \eta) \wedge$
$E^2_{Au} tag(tge, tgr, o, \eta) \wedge$
$E^1_{Au} tag(tge, tgr, o, \eta) \wedge$
$E^0_{Au} tag(tge, tgr, o, \eta)$ [By $\models$]

17.5.3. $\forall j \in Au\ SN', o \models K_j tag(tge, tgr, o, \eta)$ [Since $o \in Au$ (17.1.5.)]

17.5.4. $SN', o \models K_o tag(tge, tgr, o, \eta)$

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

$\square$

B.4 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 $\mathcal{FPPF}^{\mathcal{D}}_{\text{Facebook}}$.

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. $\underline{\text{Executing } FR1 \text{ and } FP1 \text{ enabled}}$

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. $e = acceptTagRequest(acptr, tge, tgr, picture(o, \eta))$
18.1.5. We want to prove:

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

18.2. By contradiction, let us assume
18.2.1. $SN \models_C FP1(u)$ and $SN \xrightarrow{e} SN'$
18.2.2. $SN' \not\models_C FP1(u)$

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

18.4. By Def. of $FR2.2$, we have that

18.4.1. $SN, apctr \models K_{acptr} tagRequest(tge, tgr, o, \eta)$[By Def. $FR2.2$, $acptr == tge$]

18.4.2. $SN, tge \models K_{tge} tagRequest(tge, tgr, o, \eta)$

18.5. By Def. of $FR2.2$, we have that

18.5.1. $SN', tge \models C^3_{Au} tag(tge, tgr, o, \eta)$ [By $\models$]

18.5.2. $SN', tge \models E^0_{Au} tag(tge, tgr, o, \eta) \wedge$
$E^1_{Au} tag(tge, tgr, o, \eta) \wedge$
$E^2_{Au} tag(tge, tgr, o, \eta) \wedge$
$E^3_{Au} tag(tge, tgr, o, \eta)$ [By $\models$]

18.5.3. $SN', u \models E^1_{followers(u) \cup \{u\}} tweet(u, \eta)$ [By $\models$]

18.5.4. $\forall j \in Au \; SN, j \models K_j tag(tge, tgr, o, \eta)$

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

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