Formalising Privacy Policies in Social Networks

Raúl Pardo[a], Musard Balliu[a], Gerardo Schneider[b]

[a]*Department of Computer Science & Engineering, Chalmers University of Technology, Sweden*
[b]*Department of Computer Science & Engineering, University of Gothenburg, Sweden*

Abstract

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

Keywords: Social Networks, Epistemic Logic, Privacy

1. Introduction

Over the past decade, the use of Social Network Services (SNS) like Facebook and Twitter has increased to the point of becoming ubiquitous. A recent survey shows that nearly 70% of the Internet users are active on SNSs [1]. Empirical studies show that the number of privacy breaches is keeping pace with this growth

Email addresses: pardo@chalmers.se (Raúl Pardo), musard@chalmers.se (Musard Balliu), gerardo@cse.gu.se (Gerardo Schneider)

and users' requirements are much higher than the privacy guarantees offered by SNSs [2, 3, 4, 5, 6].

Motivation: In this paper we are concerned with privacy issues in SNSs. According to Boyd and Ellison [7] SNSs have three distinguishing characteristics that differentiate them from other services: i) A public profile; ii) A set of connections between users; iii) The ability for users to see certain information about others they are connected to, including meta-information such as others' connections. These features make SNSs susceptible to privacy breaches at different levels. Though the users of an SNS control much of the information disclosed about themselves, to date it is not clear whether such controls match the users' privacy intentions [8]. An additional concern is whether the privacy settings currently available in SNSs are suitable for capturing the needs of most users through a good privacy policy language. Privacy policies should also take into account that the networks *evolve*, for instance, when introducing new users or sending posts to other users. Many desirable privacy policies can already be enforced by SNSs; for instance, in Facebook users can state policies like "Only my friends can see posts on my timeline". Many other policies, however, are not possible to enforce, although they might be important from a user's privacy perspective. Again, in Facebook users can not specify privacy policies like "I do not want to be tagged in pictures by anyone other than myself" (P1) or "Nobody apart from myself can know my child's location" (P2). Although SNSs put more and more effort in improving users' privacy, the increasing amount of information that SNSs have to deal with and the continuous policy changes make this task cumbersome and hard to accomplish.

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

Our proposal: Our aim in this paper is to provide a suitable formalism for writing and reasoning about privacy policies in *dynamic* SNSs, and to enable a formal assessment on whether these policies are properly enforced by the SNSs. Our starting point is the definition of a formal framework for privacy policies consisting of: i) a generic model for social networks, ii) a knowledge-based logic to reason about the social network and privacy policies; iii) a formal language to describe privacy policies (based on the logic above).
Epistemic logic has been successfully used as a formal specification language in many settings [9, 10, 11]. Here, we advocate that this logic is natural for privacy in SNSs. We start with first-order logic to represent connections between users and enrich it with epistemic (K) and deontic (P) operators to express knowledge and permissions, respectively. For instance, if the logical formula $P_{Bob}^{Alice}tag$ specifies that "Bob is permitted to tag Alice in any picture", then we can write $[\![\neg P_{other}^{me} tag]\!]_{me}$ to model the policy P1 above. The wrapper $[\![\,]\!]_{me}$ is used to specify the owner of the privacy policy, in this case me. Similarly, if $S_{All}\, location(me)$ stands for "Someone among all users in the SNS knows my location", we can write $[\![\neg S_{All\setminus\{me\}} location(myChild)]\!]_{me}$ to specify the policy P2. Moreover, we can express more precise policies by nesting knowledge operators. Suppose Charlie is organising an event ev and he wants both Alice and Bob to participate; however, Alice will not participate if she knows that Bob is going. Then, Charlie, who definitively wants Alice to participate, can write the formula $[\![\neg K_{Alice} K_{Bob} ev]\!]_{Charlie}$ to express the policy "Alice can not know that Bob knows about the event ev".

We do not explicitly use the standard Kripke semantics to define satisfaction of a formula in our logic. Mainly, this is because we aim at providing a model that preserves the inherent structure of an SNS, whereas Kripke semantics requires a technical machinery that is far from a real model of social networks. Our framework directly captures the underlying structure of SNSs and thus brings out a faithful model of reality. On the other hand, we borrow traditional epistemic logic axiomatisations to define a *deductive engine* which determines the knowledge of the users. This enables us to, firstly, reuse existing theorem provers for checking whether a user knows some information, and secondly, choose weaker axiomatisations of knowledge that in turn are known to be more efficient to compute [9], thus paving the way for automated enforcement of privacy policies.

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

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

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

We are not the first to apply epistemic reasoning in the context of privacy for SNSs. A precursor of our approach is the framework proposed earlier by two of the authors in [12]. That framework only considers a *static* picture of SNSs and, besides the general template of $\mathcal{PPF}$ (see Definition 1), the two are fundamentally different. Our work redefines $\mathcal{PPF}$ entirely and introduces a novel approach to reason about the dynamic behaviour of SNSs. We use first-order structures enriched with the epistemic modality to increase the expressiveness of the logic and the policy language. We remark that the satisfaction relation uses a deductive engine over the user's knowledge base, which fixes unnecessary complications in the semantics given in [12]. We discuss the differences between our work and the work in [12] in more detail in Section 5.

2. Privacy Policy Framework

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

The first-order privacy policy framework is equipped with several components. Firstly, we define models which leverage the well-known model for SNSs, the social graph [13]. We enrich these models with the knowledge and the permission that users have in an SNS. We represent the knowledge using a first-order epistemic (knowledge-based) structure, very much in the style of interpreted systems [9], and the permissions as links between users in the graph, similar to connections. Secondly, we introduce a knowledge-based logic to reason about the properties of the model. Finally, based on the logic, we provide an expressive language to write privacy policies. Formally, we define the framework as follows:

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

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

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

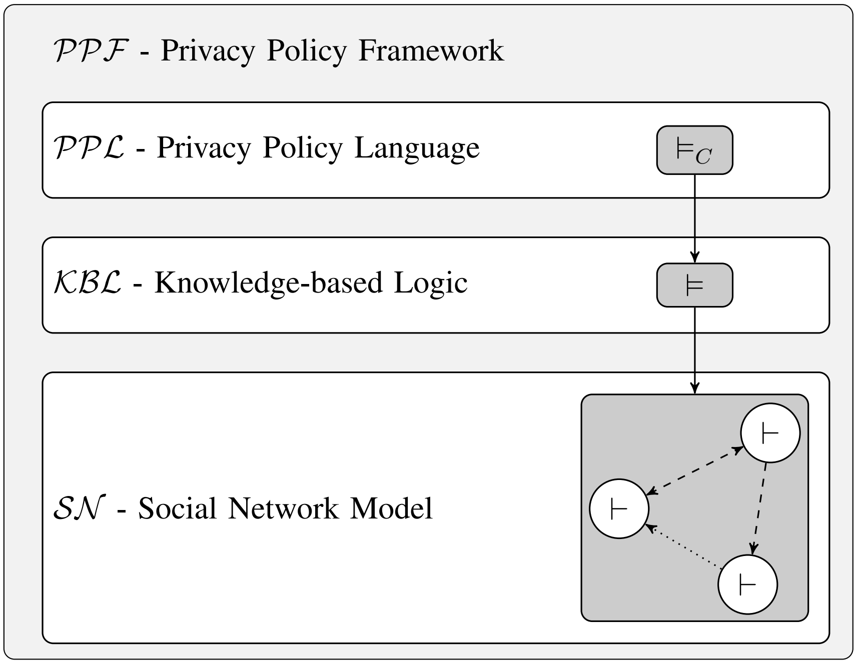
$\mathcal{PPF}$ - Privacy Policy Framework
$\mathcal{PPL}$ - Privacy Policy Language
$\vDash_C$
$\mathcal{KBL}$ - Knowledge-based Logic
$\vDash$
$\mathcal{SN}$ - Social Network Model
$\vdash$
$\vdash$
$\vdash$

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

$\vDash_C$ to $\vDash$. Social network models consist of nodes that represent users and arrows that represent connections and permissions between users. We will describe social network models in full detail in the next section. In a nutshell, each node (user) i contains a local knowledge base and a derivability relation $\vdash$, which we use to determine whether the user i knows some information φ, namely the satisfaction of formulae $K_i\varphi$. We define this by requiring that φ is derivable (using $\vdash$) from the set of facts in the user's knowledge base. The derivability relation borrows the axioms and derivation rules from the **S5** axiomatisation defined by Fagin *et al.* in [9]. By using the **S5** axiomatisation for our notion of derivability (inside the users' knowledge bases), we can obtain an axiomatisation of the $\mathcal{KBL}$ logic that corresponds to the **KD45** axiomatisation and it is sound with respect to social network models [14]. In what follows we provide a detailed description of the components and their relations.

2.1. Social Network Models

Social networks are usually modelled as graphs, where nodes represent users—referred to as *agents*—and edges represent different kinds of relationships among users, for instance, friendship or family. These graphs are traditionally called social graphs [13]. A social network model is a social graph which includes the knowledge that the agents have accumulated as a set of logic formulae in their *knowledge base*. Moreover, we model possible inferences of knowledge that the agents can make from the knowledge that they already possess. Additionally, we use new types of edges to represent certain permission that the agents may have. We write $\mathcal{AU}$ to denote a universe of agents.

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

- *Ag is a nonempty finite set of* nodes *representing the agents in the SNS.*

- *$\mathcal{A}$ is a first-order structure over the social network model. As usual, it consists of a set of domains; and a set of relations, functions and constants interpreted over their corresponding domain.*

- *$KB : Ag \to 2^{\mathcal{F}}$ is a function denoting the* knowledge base *of an agent, namely the accumulated knowledge of an agent. We write KB_i to denote $KB(i)$.*

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

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

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

The reasoning capabilities for the agents help us to make these type of inferences in the framework.

Since the knowledge of the agents is represented using epistemic logic formulae, we use standard properties of knowledge to model the reasoning capabilities of agents. We introduce a set of axioms and rules for an agent to infer new knowledge from the one present in their knowledge base. In particular, we use the knowledge axiomatisation **S5** from first-order epistemic logic [9, 15]. **S5** is a standard and widely used axiomatisation for epistemic logic. Moreover, there exist several tools to check validity of epistemic formulae for **S5**. These tools can be used to implement an enforcement mechanism based on our framework. Nevertheless, we are not restricted to **S5**. Given the modularity of our approach, we could consider other axiomatisations.

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

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

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

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

Definition 3 (First-Order **S5** [9, 15])**.** *Given the formulae φ and ψ written in $\mathcal{L}_n$ and some agent i, the axiom system **S5** consists of the following axioms and derivation rules:*

Axioms

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

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

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

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

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

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

Derivation rules

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

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

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

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

Definition 4 ([15])**.** *A* derivation *of a formula $\varphi \in \mathcal{L}_n$ is a finite sequence of formulae $\varphi_1, \varphi_2, \ldots, \varphi_n = \varphi$, where each φ_i, for $1 \leq i \leq n$, is either an instance of the axioms (A1, A2, A12, A3, A4 or A5) or the conclusion of one of the derivation rules of which premises have already been derived, i.e., appear as φ_j with $j < i$. Moreover when we can derive φ from a set of formulae $\{\psi_1, \psi_2, \ldots \psi_n\}$, if we take the set Γ as the conjunction of all the formulae from the previous set, $\Gamma = \psi_1 \wedge \psi_2 \wedge \ldots \wedge \psi_n$ we write $\Gamma \vdash \varphi$.*

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

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

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

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

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

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

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

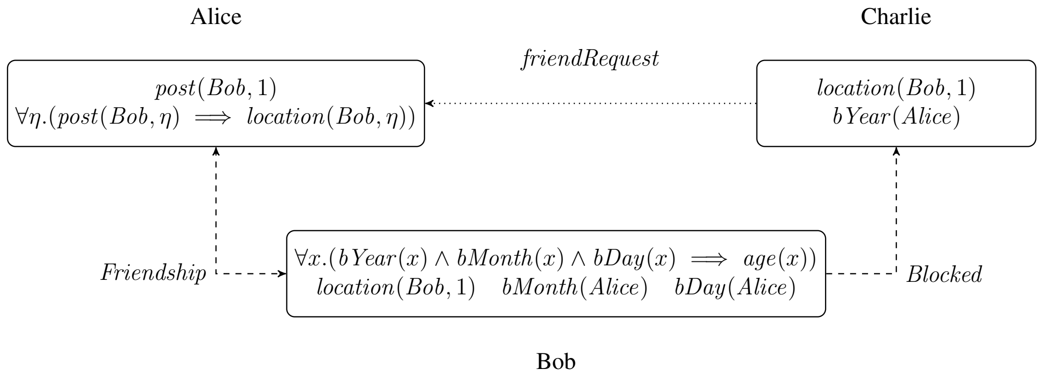
Alice
Charlie
friendRequest
$post(Bob, 1)$
$\forall \eta.(post(Bob, \eta) \implies location(Bob, \eta))$
$location(Bob, 1)$
$bYear(Alice)$
Friendship
$\forall x.(bYear(x) \wedge bMonth(x) \wedge bDay(x) \implies age(x))$
$location(Bob, 1) \quad bMonth(Alice) \quad bDay(Alice)$
Blocked
Bob

Figure 2: Example of Social Network Model

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

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

Figure 2 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 vice versa. On the other hand, it is also possible to represent unidirectional connections, as Blocked; in SN Bob has blocked Charlie. Permissions are represented using a dotted arrow. In this example, Charlie is able to send a friend request to Alice.

The predicates inside each node represent the agents' knowledge base. In this SNM, Charlie and Bob have the predicate $location(Bob, 1)$[1] in their knowledge bases, meaning that they both know location number 1 of Bob. Moreover, the

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

knowledge bases may contain not only predicates, but also other formulae. These formulae may increase the knowledge of the agents. For instance, Alice knows $location(Bob, 1)$ *implicitly. Alice can derive* $location(Bob, 1)$ *by* Modus Ponens, *from* $post(Bob, 1)$ *and* $\forall \eta . post(Bob, \eta) \implies location(Bob, \eta)$. □

2.2. Knowledge-based Logic

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

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

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

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

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

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

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

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

We choose to discriminate between predicates encoding permissions between agents, i.e., $a_n(i, j)$, predicates encoding connections between agents, i.e., $c_m(i, j)$, and other types of predicates, e.g. $p(\overrightarrow{t})$, in order to stay as close as possible to the social network models. $\mathcal{F}_{\mathcal{KBL}}$ will represent the set of all well-formed formulae of $\mathcal{KBL}$ according to the category φ above. The epistemic modalities stand for: $K_i\varphi$, agent i knows φ; $E_G\varphi$, everyone in the group G knows φ; $S_G\varphi$, someone in

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

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

the group G knows φ; $D_G\varphi$, φ is distributed knowledge in the group G; $C_G\varphi$, φ is common knowledge in the group G. We use the following operators as syntactic sugar in the logic $\mathcal{KBL}$: $P_i^j a_n := a_n(i,j)$, agent i is permitted to execute action a_n to agent j; $SP_G^j a_n := \bigvee_{i \in G} a_n(i,j)$, at least one agent in G is permitted to execute action a to agent j; $GP_G^j a_n := \bigwedge_{i \in G} a_n(i,j)$, all agents in G are permitted to execute action a to agent j. When we write "*agent i is permitted to execute action a_n to agent j*", it means that agent i allows j to perform an action a_n which directly involves i, e.g. $P_{Bob}^{Alice} friendRequest$ would mean that Bob is allowed to send a friend request to Alice. We define E_G^{k+1} as $E_G E_G^k\varphi$, where $E_G^0\varphi$ is equal to φ. The logical operators $\rightarrow$ and $\vee$ are defined in terms of $\wedge$ and $\neg$ as usual.

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

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

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

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

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

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

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

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

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

$$SN \vDash K_{\mathit{Alice}}\mathit{location}(Bob, 1) \tag{4}$$

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

$$\forall \eta.\mathit{post}(Bob, \eta) \implies \mathit{location}(Bob, \eta) \tag{5}$$

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

$$\mathit{post}(Bob, 1) \implies \mathit{location}(Bob, 1) \tag{6}$$

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

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

Finally, by the definition of Cl (Definition 5), modus ponens *can by applied for (6) and (3) to derive $\mathit{location}(Bob, 1)$, i.e. $\mathit{location}(Bob, 1) \in \mathit{Cl}(KB_{\mathit{Alice}})$ and therefore (4) holds.* □

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

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

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

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

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

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

it implies that

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

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

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

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

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

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

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

□

Properties of the framework. The logic $\mathcal{KBL}$ leverages the deductive engine by applying the axioms and derivation rules from the axiomatisation **S5** to infer new knowledge. We remark that the same axiomatisation can not be used for $\mathcal{KBL}$, since the predicates, e.g. connection and permission, are interpreted differently depending on whether or not they occur inside a knowledge modality. For instance, satisfaction of the connection predicate $\mathit{friendship}(Alice, Bob)$ will only depend on the condition $(Alice, Bob) \in A_{\mathit{Friendship}}$. Nonetheless, checking the formula $K_{\mathit{Alice}}\mathit{friendship}(Alice, Bob)$ requires that $\mathit{friendship}(Alice, Bob) \in Cl(KB_{\mathit{Alice}})$. This is in line with the fact that agents may know facts that are not true in the real world. As a result, this unconventional interpretation of predicates prevents us from using axiomatisations defined for classical epistemic logic. However, when checking if a formula is in the knowledge base of an agent, all predicates are treated equally, even when they are connection or permission predicates. Hence the individual knowledge of each agent in SNMs can be modelled using a classical Kripke model, meaning that it can be seen as a set of formulae in $\mathcal{L}_n$. Because of this, we assume that they can infer new knowledge using the axiomatisation **S5**, which is sound and complete for classical epistemic logics [9].

The example above shows that we are concern with agents' belief instead of knowledge. In fact, we can use the **KD45** axiomatisation for belief [9] to check properties of the logic $\mathcal{KBL}$ [14]. Intuitively, we can think of SNMs as models that combine two logics. On the one hand, we use $\mathcal{KBL}$ to reason about the global knowledge and permission of the SNS. On the other hand, agents can have their local knowledge represented using $\mathcal{L}_n$ and use **S5** to infer new knowledge. The $\mathcal{KBL}$ logic is closely related to traditional Kripke models as discussed in Section 5.

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

2.3. The Privacy Policy Language

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

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

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

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

- *Direct restrictions* - $[\![\neg\alpha]\!]_i$ These are restrictions which allow users to explicitly specify the audience which has no access to some piece of information or who is permitted to execute an action. For instance, in $\mathcal{PPL}$ agent i can write $[\![\neg S_{\{m,n,o\}}p(\overrightarrow{t})]\!]_i$, 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 (see Example 5).

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

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

The intuitive meaning of this policy is that if a user $i \in Ag$ creates an event $event(i, descp)$ (where $descp$ is the description of the event) and she gives permission to join it (the action of joining the event is represented by $join_{event(i)}$)

to a certain group of people, then the event cannot be accessed by people other than the ones who are allowed to join it. Similarly, a user can choose to limit the event's audience to her friends only. This can be expressed in $\mathcal{PPL}$ as

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

which is also equivalent to

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

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

2.4. Instantiation of the framework

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

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

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

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

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

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

The *admissibility* condition $\mathbb{AC}_{\mathcal{S}}$ specifies the *generic structure* of privacy policies for a particular instantiation (see Definition 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 $\mathbb{AC}_{\mathcal{S}}$. Independently of the admissibility condition, we assume that all privacy policies are consistent. For simplicity, when no confusion arises, we will not specify the subindex $\mathcal{S}$ in the instantiation. Also, as mentioned before, the deductive engine of the *knowledge base*, KB, is extended with the assumptions $\mathbb{A}_{\mathcal{S}}$ in the instantiation.

2.5. Instantiation of Twitter

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

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

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

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

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

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

Actions. The actions are defined as:

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

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

Constraints over privacy policies (Admissibility condition). In $\mathcal{FPPF}_{\text{Twitter}}$ we do not define constraints for the privacy policies *per se*. Instead we describe a schema composed by the generic structure of the privacy policies that users in Twitter can write. The schema is based on the set of Twitter privacy policies presented in [12], which was shown to express all the policies that Twitter offers in its privacy settings section nowadays.

Definition 12. *Given $u \in Ag$ and $\eta \in \mathbb{N}$; the generic structure of the privacy policies for Twitter is as follows:*

$P1(u) = [\![\neg S_{Ag\setminus followers(u)\setminus\{u\}}\ tweet(u,\eta)]\!]_u$ *- Only u's followers can access her tweets.*

$P2(u) = [\![\neg S_{Ag\setminus followers(u)\setminus\{u\}} retweet(u,tu,\eta)]\!]_u$ *- Only u's followers can access her retweets.*

$P3(u) = [\![\neg S_{Ag\setminus\{u\}}\ location(u,\eta)]\!]_u$ *- No tweet by u contains her location.*

$P4(u) = \forall i.[\![\neg K_i\ (email(u) \vee phone(u)) \implies \neg P_i^u\ accessProfRec]\!]_u$ *- No user i can receive a recommendation to follow u, unless i knows u's email or phone number.*

$P5(u) = [\![\neg SP^u_{Advertisers}\ sendAd]\!]_u$ *- No advertisement companies can send ads to user u.*

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

Auxiliary functions. The set of auxiliary functions consists of:

- $followers : Ag_{Twitter} \rightarrow 2^{Ag_{Twitter}}$ - This function returns the followers of a given user, i.e. for $u \in Ag_{\text{Twitter}}$, $followers(u) = \{\, i \mid (i,u) \in C_{Follower}\}$.

- $state : Ag_{Twitter} \rightarrow St$ - This function returns the state of a user's account, which can be *public* or *private*. For $u \in Ag_{\text{Twitter}}$, it returns *private* if the policies $P1, P2 \in \pi_u$ and *public* otherwise.

- $inclocation : Ag_{Twitter} \rightarrow Bool$ - This function returns the user's preference for revealing the location with the tweet. For $u \in Ag_{\text{Twitter}}$ it returns $false$ if the policy $P3 \in \pi_u$, and $true$ otherwise.

- $beingReco : Ag_{Twitter} \rightarrow Bool$ - This function returns the user's preference about being recommended to be followed by other users who have access her email or phone number. For $u \in Ag_{\text{Twitter}}$ it returns $false$ if the policy $P4 \in \pi_u$, and $true$ otherwise.

- $getTweetInfo : Ag_{Twitter} \times \mathbb{N} \rightarrow 2^{\mathcal{P}_{Twitter}}$ - This function extracts information from a given tweet, for instance, the location ($location(o,\eta)$), the users mentioned in the tweet ($mention(u_1,o,\eta)\ \ldots\ mention(u_m,o,\eta)$), and the attached pictures ($picture(o,\eta,1) \ldots picture(o,\eta,j)$), where $m, j \in \mathbb{N}$ are indexes. This information is returned as a set of predicates.

- $audience : \mathcal{P}_{\mathit{Twitter}} \rightarrow 2^{Ag_{\mathit{Twitter}}}$ - This function returns the audience of some piece of information, i.e. the agents who know that information. Given $p(\overrightarrow{t}) \in \mathcal{P}_{\text{Twitter}}$, $audience(p(\overrightarrow{t})) = \{\ i \mid SN \vDash K_i p(\overrightarrow{t})\}$

- $info : Ag_{\mathit{Twitter}} \rightarrow 2^{\mathcal{P}_{\mathit{Twitter}}}$ - This function returns all the information of a given agent. Given an agent $u \in Ag_{\text{Twitter}}$, $info(u) = \{p(u, \overrightarrow{t}) | p(u, \overrightarrow{t}) \in KB_u\}$.

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

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

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

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

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

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

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

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

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

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

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

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

3. Privacy Policies in Dynamic SNS

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

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

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

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

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

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

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

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

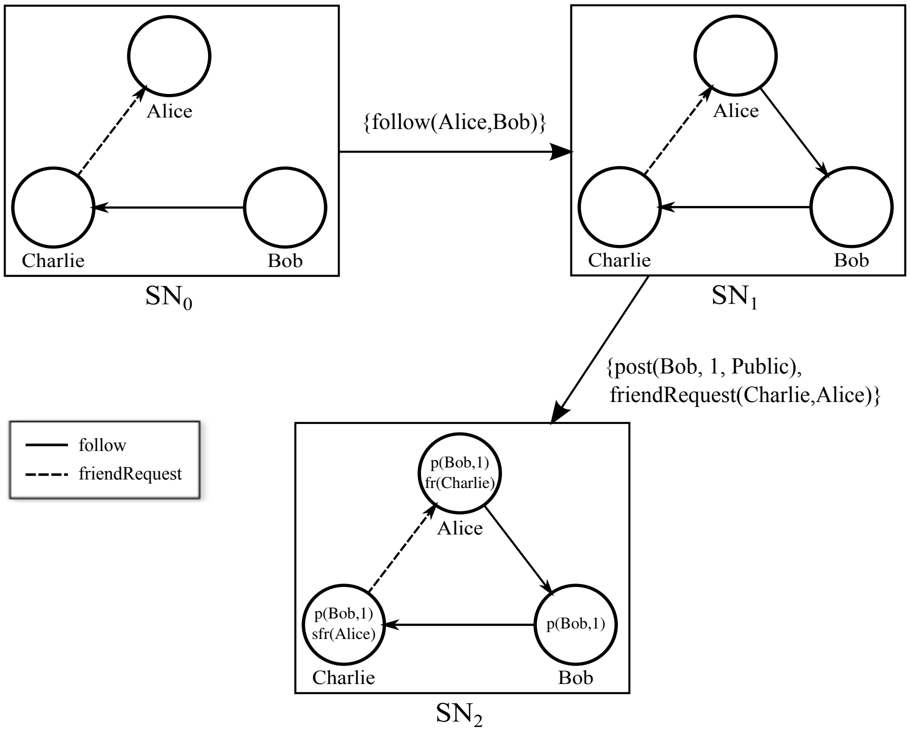

Figure 3: Example of SNS Labelled Transition System

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

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

$$SN_1 \vDash P_{Charlie}^{Alice} friendRequest$$

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

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

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

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

Definition 15. *A* dynamic $\mathcal{FPPF}$ instantiation*, denoted as* $\mathcal{FPPF}_{\mathcal{S}}^{\mathcal{D}}$*, 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:*

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

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

3.3. Operational Semantics Rules for SNS

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

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

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

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

Sometimes SNSs release information by making a message available to a group of users, e.g. tweets on Twitter or posts on Facebook. In dynamic epistemic logic this type of event is known as *public announcement* [16]. The result of performing a public announcement is that the disclosed information becomes *common knowledge* to the group of agents which are the audience of that announcement. We use common knowledge to accurately model what is known for everyone in a group.

Epistemic

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

Topological

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

Policy

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

Hybrid

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

Table 3: Generic structure of the operational semantics rules

Topological. These rules only affect the topology of SNMs. The social topology represents the elements of an SNM which come from the social graph. Therefore these rules update the sets Ag and C_i of an SNM. Using topological rules we can model the addition or removal of users and the relationships among them.

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

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

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

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

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

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

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

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

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

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

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

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

3.4. Dynamic Instantiation of Twitter

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

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

- $\mathit{changeStPriv}$, $\mathit{changeStPub}$ - These events model the switching between *'Private'* or *'Public'* accounts.

- $\mathit{inclLoc}$, $\mathit{notInclLoc}$ - These events represent whether the location is included or not in the tweet, respectively.

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

The event $\mathit{createProf}$ describes how the social network model changes when a new user joins the SNS, i.e., $\mathit{SN} \xrightarrow{\mathit{createProf}(u,\mathit{InitialInfo})} \mathit{SN}'$ for $\mathit{SN}, \mathit{SN}' \in \mathcal{SN}_{\text{Twitter}}$, $u \in \mathit{Ag}$ and $\mathit{InitialInfo} \subseteq \mathcal{F}_{\mathcal{KBL}}$ (representing the initial set of information that users provide in Twitter). Rule (T2.1) consists of one condition, which if satisfied, leads to four consequences. The condition $u \notin \mathit{Ag}$ 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, $\mathit{Ag}' = \mathit{Ag} \cup \{u\}$ (where $\mathit{Ag}' \in \mathit{SN}'$), specifies that the new user is added to the SNM. Secondly, $\mathit{KB}'_i = \mathit{InitialInfo}$, represents that in the new SNM SN', the user knows all the information she provided when signing up. Moreover the user is able to access her own profile as represented by $A'_{\mathit{accessProf}} = A_{\mathit{accessProf}} \cup \{(u, u)\}$. Finally, $\forall j \in \mathit{Advertisers}\ A'_{\mathit{sendAd}} = A_{\mathit{sendAd}} \cup \{(u, j)\}$, models the set of advertisers, $\mathit{Advertisers} \subseteq \mathit{Ag}$, who can send advertisements to the user.

In general, an event may give rise to more than one operational semantics rule. tweet is an example of such an event (see Table 4). It is composed by 4 rules, which determine its behaviour depending on certain conditions. These conditions consider whether a user has protected her tweets and whether she allows her location to be included in her tweets. Since the policies can be either activated or deactivated, this leads to four different social network models after its execution. Suppose that $\mathit{SN} \xrightarrow{\mathit{tweet}(u,\mathit{TweetInfo})} \mathit{SN}'$ for $\mathit{SN}, \mathit{SN}' \in \mathcal{SN}_{\text{Twitter}}$, $u \in \mathit{Ag}$ and $\mathit{TweetInfo} \in 2^{\mathcal{P}_{\text{Twitter}}}$ (representing the information disclosed in the tweet, i.e., location of the tweet, mentions, pictures, etc). In the first line of all the rules for tweet we specify what will be the audience of the tweet. This depends on the type of the account of the user who is tweeting. If the state of the user's account is *'Public'*, then the tweet will be disclosed to her followers and to the people men-

Tweet - T1

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

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

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

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

Create Profile - T2

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

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

tioned in the tweet, $\mathit{followers}(u) \cup \{u\} \cup \{v \mid \mathit{mention}(v, u, \eta) \in \mathit{TweetInfo}\}$ (rules (T1.1) and (T1.3)). Otherwise, the audience is restricted to only her followers

$\mathit{followers}(u) \cup \{u\}$ (rules (T1.2) and (T1.4)). If the tweet location is deactivated, $\mathit{inclocation}(u) == \mathit{false}$, then the rules contain one extra condition which explicitly requires that the location should not be part of the information disclosed in the tweet, $\mathit{location}(u, \eta) \notin \mathit{TweetInfo}$ (rules (T1.2) and (T1.3)). As a result, all the formulae describing the tweet information become common knowledge among the agents of the audience, $\forall \varphi \in \mathit{TweetInfo}, \forall i \in Au\ K_i' = K_i \cup \{C_{Au}\varphi\}$.

The reader may wonder why the audience of a tweet is not all Twitter users when the profile of the tweet's owner is public. The reason is because we want to model the exact behaviour of the SNS. In Twitter when a user (with a public profile) tweets a message, this message is shown in her followers' timeline. Additionally, since the profile is public, any other user (who is not following her) can check all her tweets. This is modelled with the event $\mathit{accessProf}$. 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 Appendix A.3.

4. Proving Privacy in Social Networks

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

4.1. Does an SNS preserve privacy?

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

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

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

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

4.2. Privacy in Twitter

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

Theorem 1. *Twitter is privacy-preserving.*

Proof sketch: We check that the execution of none of the events in EVT_{Twitter} can violate any of the privacy policies in Π_{Twitter} by considering all possible combinations of events and privacy policies (i.e. ensuring that Definition 16 holds). Here we only show the case when $tweet$ (see Table 4) is executed and the privacy policy $P1(u) = [\![\neg S_{Ag \setminus followers(u) \setminus \{u\}}\ tweet(u, \eta)]\!]_u$ is activated. We follow the same strategy for the remaining cases (see 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:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

4.3. What about Facebook?

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

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

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

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

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

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

The previous elements constitute the static part of our (reduced) instantiation of Facebook, $\mathcal{FPPF}_{\text{FB-Tag}}$. In order to model the behaviour of the tagging event, we extend $\mathcal{FPPF}_{\text{FB-Tag}}$ with the operational semantics rules for the events $\{\mathit{tag}, \mathit{acceptTagRequest}\} \subseteq \mathit{EVT}_{\text{FB-Tag}}$ as specified in Table 5, which defines $\mathcal{FPPF}^{\mathcal{D}}_{\text{FB-Tag}}$. The intuition behind the operational semantics rules is as follows.

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

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

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

The event $\mathit{acceptTagRequest}(\mathit{acptr}\ \mathit{tge},\ \mathit{tgr},\ \mathit{picture}(o,\ \eta))$ 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., $\mathit{acptr} == o$,

Tag - FR1

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

Accept tag request - FR2

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

Table 5: Tagging - Operational Semantics of Facebook

therefore it is required to check that the user accepting the tag has access to the tag request,

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

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

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

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

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

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

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

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

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

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

As in Twitter, the following holds:

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

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 $\mathit{tag}(tgr, tge, o, \eta)$ in order to tag tge in $\mathit{picture}(o, \eta)$ where $o \in Ag$ and $\eta \in \mathbb{N}$. The owner of $\mathit{picture}(o, \eta)$ is o. Assume a social network model, SN, where it holds that $\mathit{tagRequest}(tge, tgr, o, \eta) \in KB(o)$. In order for $\mathcal{FPPF}^{\mathcal{D}}_{\text{FB-Tag}}$ to preserve privacy it must hold that if $SN \vDash_C$ *FP1*(tge) and $SN \xrightarrow{\mathit{acceptTagRequest}(o, tge, tgr, \mathit{picture}(o,\eta))} SN'$, then $SN' \vDash_C$ *FP1*(tge) where $SN, SN' \in \mathcal{SN}_{\text{FB-Tag}}$.

Since tge was not tagged before the execution of *FR2* we know that $SN \vDash_C$ *FP1*(tge). Also since $\mathit{tagRequest}(tge, tgr, o, \eta) \in KB(o)$ and $acptr == o$ we know that *FR2* can be executed. By the definition of *FR2*, we know that $SN' \vDash E_{Au} \mathit{tag}(tge, o, o, \eta)$, hence $SN' \nvDash_C$ *FP1*(tge), which contradicts our claim $SN' \vDash_C$ *FP1*(tge) and therefore $\mathcal{FPPF}^{\mathcal{D}}_{\text{FB-Tag}}$ 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 user without their approval in any of her pictures. In this instantiation, since there are only two operational semantics rules, it is easy to discuss a possible modification in the rules so that *FP1* is supported.

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

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

Tag - FR1

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

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

Accept tag request - FR2

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

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

Table 6: New Tagging rules supporting FP1

is replaced with

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

The resulting operational semantics rules are presented in table 6.

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

Lemma 3. *Facebook is privacy-preserving.*

Proof sketch: We consider all possible rules that can be executed and show that none of them will violate *FP1*, which is the only policy available in the instantiation $\mathcal{FPPF}^{\mathcal{D}}_{\text{FB-Tag}}$. The only rule that can violate *FP1* is *FR2* (specifically the case *FR2.2*). Due to the similarity to the proof for Theorem 1 we omit the details here, but we follow the same strategy, i.e. we show by contradiction that *FP1* cannot be violated. We refer the reader to Appendix B.4 for the complete proof. □

5. Discussion and Related Work

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

5.1. Epistemic Logic and SNSs

In the past, epistemic logic has been widely used for analysing security and privacy properties in multi-agent systems (MAS). Traditionally the evolution of knowledge in epistemic logic is modelled by means of runs and events, in the "run-and-systems" framework, known as *Interpreted Systems* [9].

Halpern *et al.* [11] use Interpreted Systems to formalise the notion of secrecy in MAS. They redefine the possibilistic and probabilistic security properties in epistemic logic, in the form of a modal operator which allows them to reason about knowledge, nondeterminism and probability together. Interpreted Systems also appear in [17], where Balliu presents a knowledge-based account to specify information flow conditions in a distributed setting. The main advantage of this approach is that it is able to express complex policies based on epistemic logic. One of the main drawbacks of Interpreted Systems is the high complexity of the model-checking. Nevertheless it has been studied how to implement efficient model-checkers which make it possible to verify properties of real systems. For instance, MCK [18] and MCMAS [19] are state of the art model checkers for temporal-epistemic logics based on Interpreted systems. They have successfully been used to verify security properties for several cryptographic protocols. We are not aware of any specific use for verifying privacy policies.

Interpreted systems allow to represent the knowledge at different points in time. There is no formal definition of the events that can be executed in order to specify how knowledge evolves. Instead they require a description of the protocol which models the evolution of knowledge. *Dynamic Epistemic Logic* (DEL) provides a basis for operations on knowledge evolution in epistemic logic [16, 20]. 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 [21] 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 [22], DEL has been used to study, by means of a formal technique, the effect "Revolt or Stay-at-Home" in SNSs. This effect represents how the fact of knowing how many people (or agents) are going to revolt could influence our own decision to revolt or stay at home.

DEL turns out to be not well-suited to our setting. Firstly, because it is based in the classical Kripke semantics for epistemic logic [9]. As we describe in Section 5.3, there are properties of knowledge that need to be further studied before we can encode SNMs in Kripke structures. Secondly, DEL is only used for modelling the evolution of knowledge, in our framework apart from epistemic rules, we allow for topological, policy and hybrid rules. Finally, and most importantly, the events defined in DEL are not equipped with conditions, i.e. the execution of events does not depend on the knowledge of the agents. By contrast, the execution of events in SNSs depends not only on the agents' knowledge, but also other network-dependent factors. As described in Section 3.3, we use the premises of the rules when stating the conditions for each event.

5.2. Relationship-based Access Control

Currently SNS users share their resources by using the so called Relationship-Based Access Control (ReBAC). This paradigm gives access to user resources depending on her relationships with the owner of the resource. Fong *et al.* introduce a formalism that aims at providing a better understating of ReBAC [23, 24]. They develop a general formalism which can be instantiated, first in mono-relational social networks, e.g. Facebook-like networks where the relationship between agents is friendship, and later in a more general setting, with poly-relational social networks where the type of the relationship is also taken into account (e.g. patientphysician, parent-child). In addition, they introduce the notion of *access contexts*, defined as a hierarchy of contexts to enable an inheritance mechanism of relationships. Hence the access to the resources also offers the possibility of articulating relationships between users depending on the access context. The audience of the resources is defined by means of ReBAC policies. In [25] Bruns *et al.* provide a language based on a hybrid logic which extends Fong's work and supports interesting policy idioms.

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

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

The main difference between the semantics of $\mathcal{FPPF}$ and First-Order Epistemic Logic (FOEL) is the way knowledge is interpreted. SNMs "store" in each node a set of $\mathcal{F}_{\mathcal{KBL}}$ formulae that represent what an agent knows, namely the knowledge base of the agent. On the contrary, in relational Kripke structures, the *uncertainty* of the agents is modelled by means of a binary relation ($\mathcal{K}$) among states in the Kripke structure [9, 15]. The binary relation represents all the states that an agent considers possible. If a formula is true in all those states, then the agent knows that formula.

Nevertheless, this does not mean that the two models are complementary. In [9] Fagin *et al.* show how to construct *knowledge bases* for systems consisting of several agents by using *knowledge-based programming*. They define the state of an agent as a tuple containing all formulae the agent knows at a particular point in time. In addition to this information, the SNMs contain additional information regarding permissions and connections between users. As a matter of fact, we have shown that given a formula φ which characterises the knowledge, permission and connections of all agents in the SNM, a relational Kripke structure can be constructed containing the same information. Concretely, the canonical Kripke structure [9] resulting from φ can be built [14].

6. Conclusions and Future Work

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

Enforcement

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

On the other hand, we would like to provide a runtime enforcement mechanism for SNSs under consideration. The main advantage of this approach is that it is partially independent of the implementation of the SNS. It only tracks the require information so that it can ensure that no privacy policy is violated. We are currently studying how to extract a monitor from the specification of the privacy policies, which would run in parallel with the SNS. The monitor checks that the privacy policies of the users are not violated as they execute events. To avoid the bottleneck of a centralised algorithm, we are considering a distributed implementation. We are already implementing $\mathcal{FPPF}$ in an open source SNS called Diaspora* [26, 27] to show the practicality of our approach.

Privacy Policies and Time

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

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

Acknowledgements

This research was supported by the Swedish funding agency SSF under the grant *DataBIn: Data Driven Secure Business Intelligence*. We would like to thank Deepak Garg for his comments on earlier versions of this paper.

References

[1] A. Lenhart, K. Purcell, A. Smith, K. Zickuhr, Social media & mobile internet use among teens and young adults. millennials., Pew internet & American life project.

[2] L. Bauer, L. F. Cranor, S. Komanduri, M. L. Mazurek, M. K. Reiter, M. Sleeper, B. Ur, The post anachronism: the temporal dimension of Facebook privacy, in: WPES'13, ACM, 2013, pp. 1–12.

[3] M. Madejski, M. Johnson, S. Bellovin, A study of privacy settings errors in an online social network, in: PERCOM Workshops'12, IEEE, 2012, pp. 340–345.

[4] M. Johnson, S. Egelman, S. M. Bellovin, Facebook and privacy: It's complicated, in: SOUPS '12, ACM, New York, NY, USA, 2012, pp. 9:1–9:15.

[5] Y. Liu, K. P. Gummadi, B. Krishnamurthy, A. Mislove, Analyzing Facebook privacy settings: User expectations vs. reality, in: IMC '11, ACM, 2011, pp. 61–70.

[6] M. Madejski, M. L. Johnson, S. M. Bellovin, The failure of online social network privacy settings, Tech. Rep. CUCS-010-11, Columbia University Computer Science Technical Reports (2011).

[7] B. Boyd, N. B. Ellison, Social network sites: Definition, history, and scholarship, Journal of Computer-Mediated Communication 13 (2008) 210–230.

[8] N. B. Ellison, J. Vitak, C. Steinfield, R. Gray, C. Lampe, Privacy Online, Springer, 2011, Ch. 3: Negotiating Privacy Concerns and Social Capital Needs in a Social Media Environment, pp. 19–32.

[9] R. Fagin, J. Y. Halpern, Y. Moses, M. Y. Vardi, Reasoning about knowledge, Vol. 4, MIT press Cambridge, 2003.

[10] R. Pucella, Knowledge and security, arXiv preprint arXiv:1305.0876.

[11] J. Y. Halpern, K. R. O'Neill, Secrecy in multiagent systems, ACM Transactions on Information and System Security (TISSEC) 12 (1) (2008) 5.

[12] R. Pardo, G. Schneider, A formal privacy policy framework for social networks, in: SEFM'14, Vol. 8702 of LNCS, Springer, 2014, pp. 378–392.

[13] K. Erciyes, Complex Networks: An Algorithmic Perspective, 1st Edition, CRC Press, Inc., Boca Raton, FL, USA, 2014.

[14] R. Pardo, G. Schneider, Model checking social network models, Tech. rep., Chalmers University of Technology, URL: http://www.cse.chalmers.se/ pardo/papers/relation-ppf-kripke.pdf (2016).

[15] J.-J. C. Meyer, W. V. D. Hoek, Epistemic Logic for AI and Computer Science, Cambridge University Press, New York, NY, USA, 1995.

[16] J. van Benthem, J. van Eijck, B. Kooi, Logics of communication and change, Information and computation 204 (11) (2006) 1620–1662.

[17] M. Balliu, A logic for information flow analysis of distributed programs, in: Secure IT Systems, Springer, 2013, pp. 84–99.

[18] P. Gammie, R. van der Meyden, Mck: Model checking the logic of knowledge, in: CAV'04, Vol. 3114 of LNCS, Springer Berlin Heidelberg, 2004, pp. 479–483.

[19] A. Lomuscio, H. Qu, F. Raimondi, MCMAS: A model checker for the verification of multi-agent systems, in: CAV'09, Vol. 5643 of LNCS, Springer, 2009, pp. 682–688.

[20] J. Plaza, Logics of public communications, Synthese 158 (2) (2007) 165–179.

[21] J. Seligman, F. Liu, P. Girard, Facebook and the epistemic logic of friendship, arXiv preprint arXiv:1310.6440 (2013) 229–238.

[22] J. Ruan, M. Thielscher, A logic for knowledge flow in social networks, in: AI 2011: Advances in Artificial Intelligence, Springer, 2011, pp. 511–520.

[23] P. W. Fong, M. Anwar, Z. Zhao, A privacy preservation model for Facebook-style social network systems, in: ESORICS'09, Vol. 5789 of LNCS, Springer, 2009, pp. 303–320.

[24] P. W. Fong, Relationship-based access control: Protection model and policy language, in: CODASPY'11, ACM, 2011, pp. 191–202.

[25] G. Bruns, P. W. Fong, I. Siahaan, M. Huth, Relationship-based access control: its expression and enforcement through hybrid logic, in: CODASPY'12, ACM, 2012, pp. 117–124.

[26] Diaspora*, https://diasporafoundation.org/, accessed: 2016.

[27] $\mathcal{PPF}$ Diaspora*., Test pod: https://ppf-diaspora.raulpardo.org. Code: https://github.com/ raulpardo/ppf-diaspora, accessed: 2016.

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.

Appendix 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 a user can block other users. Not allowing to follow her, and can revert this decision.

- $showReco$ - Twitter shows a selection of recommended-to-follow user recommendations to other users, when the email or the phone number of the recommended user is known by the one to whom the recommendation is shown.

- $showAdv$ - This event models the action of a company sending an advertisement to a concrete user.

- $allowAdv$, $disallowAdv$ - A user can (dis)allow a company from sending advertisement. These events model the activation and deactivation of this permission.

- $changeStPriv$, $changeStPub$ - These events model the switching between *'Private'* or *'Public'* accounts.

- $inclLoc$, $notInclLoc$ - These events represent whether the location is included or not in the tweet, respectively.

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

Appendix 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. Note that we only write the elements of $\mathcal{A}$ involved in the rule and we write "_" to denote the rest of the elements (see Definition 2).

Appendix A.3. Epistemic

R1 - Tweet

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

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

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

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

R2 - Retweet

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

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

R3 - Favourite

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

R4 - Access profile

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

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

R10 - Show recommendation

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

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

R11 - Show advertisment

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

Appendix A.4. Topological

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

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

R7 - Accept follow request

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

R8 - Block

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

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

R9 - Unblock

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

Appendix A.5. Policy

R14 - Change state to private

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

R15 - Change state to public

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

R16 - Include location on Tweets

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

R17 - Not include location on Tweets

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

Appendix A.6. Hybrid

R5 - Create profile

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

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

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

R12 - Allow advertisment

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

R13 - Disallow advertisement

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

Appendix B. Proofs

Appendix 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. Predicates to be disclosed $\mathit{TweetInfo} \subseteq 2^{\mathcal{P}}$ where $\mathit{tweet}(u, \eta) \in \mathit{TweetInfo}$

9.1.3. $e = \mathit{tweet}(u, \mathit{TweetInfo})$

9.1.4. We want to prove:

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

9.2. By contradiction, let us assume

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

9.2.2. $\mathit{SN}' \nvDash_C P1(u)$

9.3. By 18.2.2.

9.3.1. $\mathit{SN}' \nvDash_C P1(u)$ [Definition $\vDash_C$]

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

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

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

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. Executing $R1$ and $P3$ enabled

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

10.1.6. We want to prove:

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

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

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

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

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

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

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

$R2$ - 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:

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

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

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

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

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

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

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

12. Executing $R2$ and $P3$ enabled

12.1. Given

12.1.1. $u \in Ag$ (owner of $P3(u)$ and retweeter)

12.1.2. $tweet(tu, \eta)$ (tweet $\eta \in \mathbb{N}$ of user $tu \in Ag$)

12.1.3. $\mathit{TweetInfoAu} \subseteq Ag$ (audience of the retweeted tweet)

12.1.4. $RetweetAu \subseteq Ag$ (audience of the fact of retweeting)

12.1.5. $e = retweet(u, tweet(tu, \eta))$

12.1.6. We want to prove:

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

12.2. By contradiction, let us assume

12.2.1. $SN \vDash_C P3(u)$ and $SN \xrightarrow{e} SN'$

12.2.2. $SN' \nvDash_C P3(u)$

12.3. By 12.2.1. and $\vDash_C$

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

12.3.2. $SN' \vDash S_{Ag \setminus \{u\}} location(tu, \eta)$ [By $\vDash$]

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

12.4. By Definition of $R2$

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

12.5. By 12.2.1. and the definition of $\vDash_C$

12.5.1. $SN \vDash \neg S_{Ag \setminus \{u\}} location(tu, \eta)$ [Definition S_G]

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

12.5.3. $SN \vDash \bigwedge_{i \in Ag \setminus \{u\}} \neg K_i location(tu, \eta)$

12.6. By 12.4.1. and 12.5.3.

12.6.1. $SN' \vDash_C P3(u)$

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

$R3$ – None of the privacy policies in Definition 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 Definition 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. Executing $R4$ and $P1$ enabled

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

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

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

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

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

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

13.6. By 13.3.1. we have
13.6.1. $SN \vDash_C P1$ [By $\vDash_C$]
13.6.2. $SN \vDash \neg S_{Ag\setminus followers(acd)\setminus\{acd\}} tweet(acd, \eta)$ [By Definition S_G]

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

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

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

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

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

$R5 - R9$ – None of the privacy policies in Definition 12 can be violated by the rules $R5 - R9$

Since there is neither disclosure of information nor granting of permission it is not possible to violate any of the defined privacy policies.

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

15. Executing $R10$ and $P4$ enabled

15.1. Given
15.1.1. $r \in Ag$ (Owner of $P4(r)$, i.e. $P4(r) \in \pi_r$)
15.1.2. If $P4(r) \in \pi_r$ then the case $R10.1$ is the one which will be executed
15.1.3. $e = showReco(r, v)$
15.1.4. We want to prove:

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

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

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

15.4. By 15.1.2. and 15.2.1. and Definition of $R10$
15.4.1. If $SN \models K_v(\mathit{email}(r) \vee \mathit{phone}(r))$
15.4.1.1. $SN' \models K_v(\mathit{email}(r) \vee \mathit{phone}(r)) \wedge P_v^r acessProfRecommended$
15.4.2. If $SN \models \neg K_v(\mathit{email}(r) \vee \mathit{phone}(r))$
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. $SN \models \neg K_v(\mathit{email}(r) \vee \mathit{phone}(r)) \implies \neg P_v^r \mathit{accessProfRec}$ [By $\models$]
15.5.3. $SN \models \neg\neg K_v(\mathit{email}(r) \vee \mathit{phone}(r)) \vee \neg P_v^r \mathit{accessProfRec}$ [$\neg\neg_e$]
15.5.4. $SN \models K_v(\mathit{email}(r) \vee \mathit{phone}(r)) \vee \neg P_v^r \mathit{accessProfRec}$ [$\neg\neg_i$]
15.5.5. $SN \models \neg\neg(K_v(\mathit{email}(r) \vee \mathit{phone}(r)) \vee \neg P_v^r \mathit{accessProfRec})$ [Morgan]
15.5.6. $SN \models \neg(\neg K_v(\mathit{email}(r) \vee \mathit{phone}(r)) \wedge P_v^r \mathit{accessProfRec})$

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 Definition 12 can be violated by the rules $R11 - R13$

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

$\boldsymbol{R}14 - \boldsymbol{R}17$ – None of the privacy policies in Definition 12 can be violated by the rules $\boldsymbol{R}14 - \boldsymbol{R}17$

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. □

Appendix 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 $SN \vDash_C P6(u)$, but according to $R1$, we know that all users in the audience of the tweet will learn $mention(u, o, \eta)$ and $location(o, \eta)$ and therefore $SN' \nvDash_C P6(u)$.

Proof.

16. Executing R1 and P6 activated

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

16.1.5. We want to prove

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

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

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

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

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

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

□

Appendix 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 $\mathcal{FPPF}^{\mathcal{D}}_{\text{FB-Tag}}$) a user can be tagged without approving herself the tag.

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

$FP1(tge)$ and $SN \xrightarrow{acceptTagRequest(o,tge,tgr,picture(o,\eta))} SN'$ where $SN, SN' \in \mathcal{SN}_{\text{FB-Tag}}$ then $SN' \models_C FP1(tge)$.

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

Proof.

17. Executing FR1 and FP1 activated

17.1. Given
17.1.1. User $tge \in Ag$ such that $SN \models_C FP1(tge)$
17.1.2. User $o \in Ag$ such that $tge \mathrel{!}= o$
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. $SN' \models \forall o.\forall t.\forall \eta.(K_{tge} tagRequest(t, tge, o, \eta) \vee \neg S_{Ag} tag(tge, t, o, \eta))$ [By $\neg\neg_i$]
17.2.4. $SN' \models \forall o.\forall t.\forall \eta.\neg\neg(K_{tge} tagRequest(t, tge, o, \eta) \vee \neg S_{Ag} tag(tge, t, o, \eta))$ [By Morgan]
17.2.5. $SN' \models \forall o.\forall t.\forall \eta.\neg(\neg K_{tge} tagRequest(t, tge, o, \eta) \wedge S_{Ag} tag(tge, t, o, \eta))$

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

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

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

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

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

□

Appendix 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 $\mathcal{FPPF}^{\mathcal{D}}_{\text{FB-Tag}}$.

Proof.

FR1 - Tag

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

FR2 - Accept tag request

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

FR2 - FR2.2.

18. Executing $FR1$ and $FP1$ 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. $Au = \mathit{audience}(\mathit{picture}(o, \eta))$
18.1.4. $e = \mathit{acceptTagRequest}(\mathit{acptr}, \mathit{tgr}, \mathit{picture}(o, \eta))$
18.1.5. We want to prove:

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

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

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

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

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

18.6. By 18.4.2., 18.5.4. and 18.3.5. we derive a contradiction. □

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