A Formal Privacy Policy Framework for Social Networks[⋆]

Raúl Pardo[1] and Gerardo Schneider[2]

[1] Dept. of Computer Science and Engineering, Chalmers, Sweden.
[2] Dept. of Computer Science and Engineering, University of Gothenburg, Sweden
{pardo@chalmers.se, gerardo@cse.gu.se}

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

1 Introduction

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

Privacy in SN may be compromised in different ways: from direct observation of what is posted (seen by non-allowed agents), by inferring properties of data (*metadata privacy leakages*), indirectly from the topology of the SN (e.g., knowing who our friends are), to more elaborate intentional attackers such as *sniffers* or *harvesters* [6]. In this paper we are mainly concerned with the first

[⋆] Supported by the Swedish funding agency SSF under the grant *Data Driven Secure Business Intelligence*.

3 kinds of privacy issues. In order to tackle them, we look into the problem of defining a formal language for writing rich privacy policies in the context of social networks. We aim at defining a privacy policy language able to express at least the following (kinds of) policies: i) All privacy policies currently supported by existing SN like Facebook; ii) Privacy policies describing properties on attributes, i.e. not only coarse-grained properties as the fact that someone has post something, but about the content of the post itself; iii) Conditional privacy policies, which depend on the amount of current knowledge or permissions in the SN; iv) Privacy policies based on knowledge in a group of agents and distributed knowledge among several agents.

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

1. We propose a formal *privacy policy framework* consisting of: i) a generic model for social networks, formalised as a combination of hyper-graphs and Kripke structures; ii) the syntax and semantics of a knowledge-based logic to reason about the social network and privacy policies; iii) a formal language to describe privacy policies (based on the logic mentioned above), together with a conformance relation to be able to state whether a certain social network satisfies a given policy. (Section 2.)
2. We specify how the above privacy policy framework may be instantiated in order to be used in practice. (Section 3.)
3. Our definition of *instantiated privacy policy framework* allows us to model not only existing SN with their corresponding privacy policies, but also richer ones. We show the expressiveness of our approach by presenting instantiations of Twitter, Facebook, and richer privacy policies. (Section 4.)

2 Privacy Policy Framework

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

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

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

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

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

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

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

$$\Pi = \{[\![\psi_j]\!]_i \mid i \in Ag,\ j \in \{1, 2, \ldots, n_i\} \text{ and } \psi_j \in \mathcal{PPL}_{\mathcal{SN}}\}$$

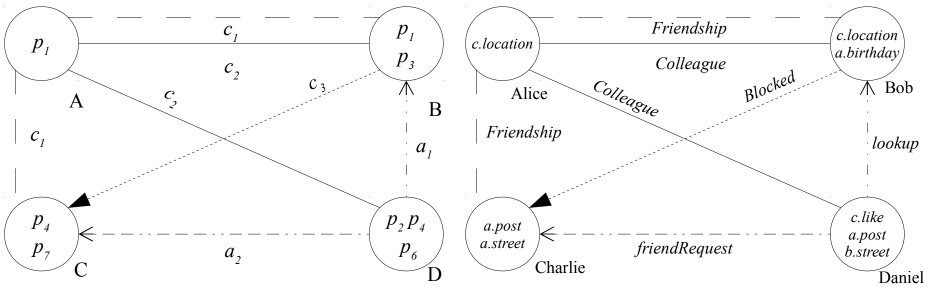
containing all the privacy policies for each agent i (there are n_i privacy policies for each agent i, if $n_i = 0$ then there is no privacy policy associated with agent i).

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

- *W is a nonempty set of* possible worlds. *Every world represents one of the agents defined in the set Ag.*
- *$\{R_i\}_{i\in\mathcal{C}}$ is a family of binary relations $R_i \subseteq W \times W$, indexed by connections. Given agents $x, y \in W$, we write xR_iy iff $(x, y) \in R_i$.*
- *$\{A_i\}_{i\in\Sigma}$ is a family of binary relations $A_i \subseteq W \times W$, indexed by actions. Given agents $x, y \in W$, we write xA_iy iff $(x, y) \in A_i$.*
- *ν is a valuation function returning the set of propositions which are true in a given world (i.e. $\nu : W \to 2^{\mathbb{P}}$).*
- *KB is a function giving the set of accumulated non-trivial knowledge for each agent, stored in what we call the* knowledge base *of the agent.* [3]
- *π is a function returning the set of privacy policies defined for a given agent (i.e. $\pi : W \to 2^{\Pi}$).* □

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

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



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

Fig. 1: Examples of social network models

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

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

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

2.2 The knowledge-based logic for social networks $\mathcal{KBL}_{\mathcal{SN}}$

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

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

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

The intuitive meaning of the modalities is as follows.

- $K_i\phi$ (Basic knowledge): Agent i knows ϕ.

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

Table 1: $\mathcal{KBL}_{\mathcal{SN}}$ satisfiability relation

- $E_G \phi$ (Everyone knows): Every agent in the group G knows ϕ.
- $S_G \phi$ (Someone knows): At least one agent in the group G knows ϕ.
- $D_G \phi$ (Distributed knowledge): ϕ is distributed knowledge in the group of agents G (i.e. the combination of individual knowledge of the agents in G).
- $P_i^j a$ (Permission): Agent i is allowed to perform action a to agent j.
- $GP_G^j a$ (Global Permission): All agents specified in G are allowed to perform action a to agent j.
- $SP_G^j a$ (Someone is Permitted): At least one agent specified in G is allowed to perform action a to agent j.

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

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

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

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

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

$$SN, A \models \neg K_B\ p_1 \wedge \neg K_C K_A\ p_1 \implies \neg SP^A_{\{B,C\}}\ a_1,$$

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

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

Lemma 1. $SN, i \models K_i\phi \wedge K_j\psi \implies D_{\{i,j\}}\phi \wedge \psi.$ □

2.3 The privacy policy language for social networks $\mathcal{PPL}_{\mathcal{SN}}$

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

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

$$\begin{aligned}
&\delta ::= \delta \wedge \delta \mid [\![\phi \implies \neg\psi]\!]_i \mid [\![\neg\psi]\!]_i\\
&\phi ::= \psi \mid \neg\psi \mid \phi \wedge \phi\\
&\psi ::= E_G\gamma \mid S_G\gamma \mid D_G\gamma \mid K_i\gamma \mid GP^j_G a \mid SP^j_G a \mid P^j_i a \mid \psi \wedge \psi.\\
&\gamma ::= p \mid \gamma \wedge \gamma
\end{aligned}$$

$SN \models_C \tau_1 \wedge \tau_2$	iff $SN \models_C \tau_1 \wedge SN \models_C \tau_2$
$SN \models_C \llbracket \neg\psi \rrbracket_i$	iff $SN, i \models \neg\psi$
$SN \models_C \llbracket \phi \implies \neg\psi \rrbracket_i$	iff $SN, i \models \phi$ then $SN \models_C \llbracket \neg\psi \rrbracket_i$

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

$\mathcal{PPL}_{\mathcal{SN}}$ may be seen as formed by a subset of formulae definable in $\mathcal{KBL}_{\mathcal{SN}}$ wrapped with the $\llbracket\ \rrbracket_i$ operator, specifying which agent has defined the privacy policy. As before, we define $\mathcal{F}_{\mathcal{PPL}}$ to be the set of $\mathcal{PPL}_{\mathcal{SN}}$ well-formed formulae defined as given by the grammar in the above definition. A basic privacy policy for an agent i, given by δ in definition 5, is either a direct restriction ($\llbracket \neg\psi \rrbracket_i$) or a conditional restriction ($\llbracket \phi \implies \neg\psi \rrbracket_i$). $\mathcal{F}^{\mathcal{C}}_{\mathcal{PPL}}$ will denote sbuformulae belonging to the syntactic category ϕ (conditions) and $\mathcal{F}^{\mathcal{R}}_{\mathcal{PPL}}$ subformulae of the syntactic category ψ (restrictions). Instead of defining a satisfaction relation for $\mathcal{PPL}_{\mathcal{SN}}$, we define the following *conformance* relation to determine when a $\mathcal{SN}$ respects a given privacy policy.

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

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

3 $\mathcal{PPF}$ instantiation

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

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

- *The set of agents* Ag*;*
- *The set of propositions* $\mathbb{P}$ *(*$p \in \mathbb{P}$ *may be given a structure);*
- *The set of connections* $\mathcal{C}$*;*
- *The set of auxiliary functions over the above connections;*
- *The set of actions* Σ*;*
- *A set of properties written in* $\mathcal{KBL}_{\mathcal{SN}}$ *(these properties may be seen as assumptions on the social network);*
- *A set of constraints over the policies defined in the language* $\mathcal{PPL}_{\mathcal{SN}}$*.* □

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

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

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

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

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

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

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

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

$$SN, i \models S_G\ u.country \wedge S_G\ u.city \wedge S_G\ u.street \implies D_G\ u.location \quad (1)$$

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

$$SN, i \models K_i(u.country \wedge u.city \wedge u.street) \implies K_i\ u.location \quad (2)$$

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

$$\text{if } SN, u \models D_G\phi \text{ and } clique(SN|_{Friendship}, G) \text{ then } SN, i \models K_i\phi. \quad (3)$$

Constraints over privacy policies. A common constraint in social networks is that agents can only write policies about their own data. In $\mathcal{PPL}_{\mathcal{SN}}$ it is possible to write $[\![\neg K_j\ u.attribute]\!]_i$ where $i, j, u \in Ag$ and $i \neq j \neq u$. This policy, defined by agent i, forbids agent j to know *attribute* from agent u. Agent i is thus constraining the accessibility of certain information about an agent other than herself. To solve this we could add the following constraint: Given an agent $i \in Ag$ and her privacy policies, $\bigwedge_{j \in 1,\ldots,n} \tau_j \in \pi(i)$, where

$\tau_j = [\![\phi]\!]_i$, if $\phi = \neg\phi'$ or $\phi = \phi'' \implies \neg\phi'$ then it is not permitted that $u.attribute \in \phi'$ for any $u \in Ag$. $u \neq i$, meaning that agents can only define policies about their own data. Likewise, users should not be able to write permission restrictions over other users. In order to address this issue we extend the previous restriction with the following: given an agent $j \in Ag$, an action $a \in \Sigma$ and the set $G \subseteq Ag$, it is not the case for $i \neq j$ that $P_u^j a$, $SP_G^j a$ or $GP_G^j a \in \phi'$. □

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

Lemma 2. *Given $u \in Ag$, if $SN, u \models D_G\ (u.country \wedge u.city \wedge u.street)$, and assuming the group of agents $G \subseteq Ag$ are all friends to each other (i.e. $clique(SN|_{Friendship}, G)$), then $SN \not\models_C [\![\neg S_{\{Ag\setminus\{u\}\}} u.location]\!]_u$.* □

4 Case Studies

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

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

4.1 Twitter privacy policies

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

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

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

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

Auxiliary functions. We define the function

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

which returns all the agents u who i is following.

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

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

- Protect my Tweets: Two cases: i) Only those in u's group of followers can see her tweets: $[\![\neg S_{\{Ag \setminus followers(u) \setminus \{u\}\}}\ u.tweet_\eta]\!]_u$; ii) Only u's followers may see her retweets: $[\![\neg S_{\{Ag \setminus followers(u) \setminus \{u\}\}} u.retweet_{tweetRef}]\!]_u$.
- Add my location to my tweets: Twitter provides the option of adding the agents' location to their tweets. The following policy specifies that nobody can see the user's locations: $[\![\neg S_{\{Ag \setminus \{u\}\}}\ u.location_\eta]\!]_u$.
- Let others find me by my email address: $[\![\neg K_i\ u.email \implies \neg P_i^u\ lookup]\!]_u$, meaning that if an agent i does not know u's email, then she is not allowed to find u by looking her up.
- Tailor ads based on information shared by ad partners: Assuming G to be the group of ads partners, the policy is defined as $[\![\neg SP_G^u\ sendAd]\!]_u$, meaning that none of the advertisement companies taking part in the system is able to send advertisements to user u.

4.2 Facebook privacy policies

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

We show here the instantiation $\mathcal{PPF}_{\text{Facebook}}$. Since we are modelling just the parts of Facebook relevant for defining privacy policies, we borrow the set $\mathcal{C}$ from the instantiation presented in Example 4. We also borrow the set of auxiliary functions and we add $friends^2(i) = \bigcup_{j \in friends(i)} friends(j)$; it allows us to write formulae about friends of friends. We extend the set Σ with the action *postTimeline* (representing posting on a user's timeline), and the actions *inviteEvent*, *inviteGroup* and *tag*, which represent sending an invitation to join an event or a group and being tagged on a picture. As for the structure of the propositions, we define the set $\mathbb{P} = \{owner.post_\eta^j, owner.like_\eta, owner.location, owner.phone, owner.email\}$ where $owner \in Ag$, $owner.post_\eta^j$ represents the posts *owner* posted on the j's timeline (e.g. $Alice.post_1^{Bob}$ is the first post of Alice in Bob's timeline), $owner.like_\eta$ are the posts *owner* has liked and $owner.location$, $owner.phone$ and $owner.email$ are, respectively, the actual location, phone and the email attributes of $owner$. Similarly to $\mathcal{PPF}_{\text{Twitter}}$, we do not specify properties for the $\mathcal{SN}$ nor restrictions over policies.

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

Who can see my stuff? In the first section Facebook enables users to set a default audience for their posts. In $\mathcal{PPF}_{\text{Facebook}}$ we can formally specify these restrictions as follows: [Public] no policy since everyone is able to access the posts; [Friends] $[\![\neg S_{\{Ag \setminus friends(u) \setminus \{u\}\}} u.post_n^j]\!]_u$; [Only me] $[\![\neg S_{\{Ag \setminus \{u\}\}} u.post_n^j]\!]_u$; [Custom] $[\![\neg S_{\{G\}} u.post_n^j]\!]_u$. The intuition behind these policies is specifying the group of agents who are not allowed to know the information about u's posts.

Who can contact me? In a second section users are provided with the possibility of deciding who can send them friend requests: [Everyone] No need of privacy policy; [Friends of Friends] $[\![\neg SP^u_{\{Ag \setminus friends^2(u)\}} request]\!]_u$; note that we specify who cannot send the friend request, which in this case are the agents who are not in the group of friends of friends.

Who can look me up? Finally, a user can be looked up by its email address or phone number. Given $a \in \{phone, email\}$, specified as [Everyone] No privacy policy is needed since; [Friends of Friends] $[\![(\neg K_i \ u.a \implies \neg P_i^u \ lookup)]\!]_u \wedge [\![(\neg SP^u_{\{Ag \setminus friends^2(u) \setminus \{u\}\}} lookup)]\!]_u$, where $i \in friends^2(u)$; [Friends] $[\![\neg K_i \ u.a \implies \neg P_i^u \ lookup]\!]_u \wedge [\![\neg SP^u_{\{Ag \setminus friends(u) \setminus \{u\}\}} lookup]\!]_u$, where $i \in friends(u)$.

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

Who can post on my timeline. Facebook offers the possibility of controlling the people allowed to write in a user's wall: [Only me] $[\![\neg SP^u_{\{Ag \setminus \{u\}\}} postTimeline]\!]_u$; [Friends] $[\![\neg SP^u_{\{Ag \setminus friends(u) \setminus \{u\}\}} postTimeline]\!]_u$.

Who can see things on my timeline? In Facebook it is possible to establish a bounded audience for the posts located in a user's wall. We formally define the privacy policies as: [Everyone] No privacy policy needed; [Friends of friends (implicitly includes friends)] $[\![\neg S_{\{Ag \setminus friends(u) \setminus friends^2(u) \setminus \{u\}\}} i.post_n^u]\!]_u$; [Friends] $[\![\neg S_{\{Ag \setminus friends(u) \setminus \{u\}\}} i.post_n^u]\!]_u$; [Only me] $[\![\neg S_{\{Ag \setminus \{u\}\}} i.post_n^u]\!]_u$; [Custom] $[\![\neg S_{\{Ag \setminus G \setminus \{u\}\}} i.post_n^u]\!]_u$.

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

4.3 More complex policies

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

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

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

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

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

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

5 Related Work

The approach we have followed in this paper has been to formally define privacy policies based on a variant of of *epistemic logic* [4], where it is possible to express the knowledge of *multi-agent systems* (MAS). One way to give semantics to the logic is to use *possible worlds semantics* (also known as *Kripke models*), where it is not explicitly represented what the agents know, but rather the *uncertainty* in their knowledge. This has the advantage of allowing to represent complex formulae about who knows what (including nesting of knowledge and other operators generalising the notion). Another way to give semantics to epistemic logic is to use *interpreted systems* which represents agent's knowledge as a set of runs (computational paths). Both ways of giving semantics come with advantages and disadvantages: Kripke models come with a heritage of fundamental techniques allowing to prove properties about the specification, while interpreted systems are quite intuitive to model MAS [8]. The common key in both approaches is modelling the uncertainty of the agent by using an equivalence relation. If one thinks about all the worlds that a given agent could consider possible in a social network system, it is easy to see that modelling them would lead to creating an enormous state space. Instead of modelling uncertainty we explicitly store what the agents know. This allows a more concise representation of the individuals' knowledge. Unlike previous work on epistemic logic, in our formalism worlds represent agents.

Moreover we explicitly model a restricted version of permission, i.e. our model explicitly shows which actions are allowed to be executed by the agents. Aucher *et al.* [1] show a different way of combining epistemic and deontic aspects in logic. They preserve the equivalence relation for epistemic properties and use an extra equivalence relation for representing permission. The logic is quite expressive but it suffers from the aforementioned state explosion problem. Furthermore the framework is defined as a mono agent system not being suitable for SN. We took their idea of combining epistemic and deontic operators in one language, but we restricted the semantic model according to the needs of SN.

In [10] Seligman *et al.* present a language based on epistemic logic, with the traditional Kripke semantics for the logic extended with a friendship relationship. By doing that they are able to reason about knowledge and friendship. Moreover they model a set of events using general dynamic dynamic logic (GDDL) by defining an *update* operation over the mentioned Kripke model. This enables the possibility of update the model as the events in the social network occur. Using GDDL they implement the concept of public and private announcement, which appear regularly in the communications among the agents. Although this approach is quite expressive, its focus is not on privacy or security issues, but in reasoning about the general knowledge of the agents. As mentioned before it comes with the price of having a immense state space and it complicates a practical implementation and the definition of an efficient (computationally speaking) model checking algorithm. Ruan and Thielscher [9] present a very similar formalism, but only public announcement is defined. Their focus is not on privacy either, but in the analysis of the "revolt or stay at home" effect, i.e. how the knowledge is spread among the agents.

There are other approaches for privacy not based on epistemic logic. One of the most interesting is Relationship-based access control (REBAC) [5]. The main difference with epistemic logic is that in REBAC the reasoning is focused on the resources own by the agents of the system. This approach is highly suitable for a practical implementation of a policy checking algorithm. On the other hand their approach would not detect certain kind of implicit knowledge flow. For instance, certain information about a user can be known after a friend of her is posting some information about both. The formalism is equipped with a formal language based on hybrid logic [2].

Datta *et al.* present in [3] the logic PrivacyLFP for defining privacy policies based on a restricted version of first-order logic (the restriction concerns that quantification over infinite values is avoided by considering only relevant instances of variables). The logic is quite expressive as it can represent things others than the kind of policies we are aiming at in this paper (their application domain being medical data). Though promising as a formalism for SN, the authors write that the logic might need to be adapted in order to be used for online social networks. To the best of our knowledge this has not been done.

6 Final Discussion

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

This paper is a first step towards a full formalisation of privacy policies for social networks. Our current and future work includes: **Adding real-time:** So far we cannot express policies with deadlines. This might be interesting in case policies are transient (e.g., "nobody is permitted to know my location during the first two weeks of May"). **Modeling dynamic networks:** The model we have of social networks is static, as well as the conformance relation between policies and the network. In practice the social network evolves, new friends come into place, others are blocked, etc. We aim at extending our formal model to capture such temporal aspect. **Adding roles and ontologies:** Agents in the SN could play different roles, e.g. individuals, companies, advertisement, etc. Providing $\mathcal{PPF}$ with the ability of detecting these roles would enhance its expressivity. **Developing an enforcing mechanism:** We have not mentioned how the policies might be enforced at runtime. We will explore how to extract a runtime monitor from the policy. Finally, we would like to explore the application of privacy-by-design [7] to a formalisation of social networks.

Acknowledgment Thanks to Bart van Delft, Pablo Buiras, and the anonymous reviewers for their useful comments on a preliminary version of this paper.

References

1. G. Aucher, G. Boella, and L. Torre. A dynamic logic for privacy compliance. *Artificial Intelligence and Law*, 19(2-3):187–231, 2011.
2. G. Bruns, P. W. Fong, I. Siahaan, and M. Huth. Relationship-based access control: its expression and enforcement through hybrid logic. In *CODASPY'12*, pages 117–124. ACM, 2012.
3. A. Datta, J. Blocki, N. Christin, H. DeYoung, D. Garg, L. Jia, D. K. Kaynar, and A. Sinha. Understanding and protecting privacy: Formal semantics and principled audit mechanisms. In *ICISS*, volume 7093 of *LNCS*, pages 1–27. Springer, 2011.
4. R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning about knowledge*, volume 4. MIT press Cambridge, 1995.
5. P. W. Fong. Relationship-based access control: Protection model and policy language. In *CODASPY'11*, pages 191–202. ACM, 2011.
6. B. Greschbach, G. Kreitz, and S. Buchegger. The devil is in the metadata - new privacy challenges in decentralised online social networks. In *PerCom Workshops*, pages 333–339. IEEE, 2012.
7. D. Le Métayer. Privacy by design: A formal framework for the analysis of architectural choices. In *CODASPY'13*, pages 95–104. ACM, 2013.
8. A. Lomuscio and M. Ryan. On the relation between interpreted systems and kripke models. In *Agents and Multi-Agent Systems Formalisms, Methodologies, and Applications*, volume 1441 of *LNCS*, pages 46–59. Springer, 1997.
9. J. Ruan and M. Thielscher. A logic for knowledge flow in social networks. In *AI 2011: Advances in Artificial Intelligence*, pages 511–520. Springer, 2011.
10. J. Seligman, F. Liu, and P. Girard. Facebook and the epistemic logic of friendship. In *TARK'13*, 2013.