Specification of Evolving Privacy Policies for Online Social Networks

Raúl Pardo, Ivana Kellyérová, César Sánchez and Gerardo Schneider
Dept. of Computer Science and Engineering,
Chalmers | University of Gothenburg, Sweden.
Email: pardo@chalmers.se, guskeliv@student.gu.se, gersch@chalmers.se
IMDEA Software Institute,
Madrid, Spain.
Email: cesar.sanchez@imdea.org

***Abstract*—*Online Social Networks* are ubiquitous, bringing not only numerous new possibilities but also big threats and challenges. Privacy is one of them. Most social networks today offer a limited set of (static) privacy settings, not being able to express *dynamic* policies. For instance, users might decide to protect their location during the night, or share information with difference audiences depending on their current position. In this paper we introduce $\mathcal{TFPPF}$, a formal framework to express, and reason about, *dynamic* (and *recurrent*) privacy policies that are activated or deactivated by context (events) or time. Besides a formal policy language ($\mathcal{TPPL}$), the framework includes a knowledge-based logic extended with (linear) temporal operators and a *learning* modality ($\mathcal{TKBL}$). Policies, and formulae in the logic, are interpreted over (timed) traces representing the evolution of the social network. We prove that checking privacy policy conformance, and the model-checking problem for $\mathcal{TKBL}$, are both decidable.**

We prove that checking privacy policy conformance, and the model-checking problem for $\mathcal{TKBL}$, are both decidable.

I. Introduction

Online Social Networks, also known as *Social Networking Sites* or *Social Network Services*, have exploded in popularity in the last years. Over the past decade, the use of Facebook [2] and Twitter [5], just to mention two of the most popular ones, has increased at the point of becoming ubiquitous. Nearly 70% of the Internet users are active on social networks as shown by a recent survey [12], and this number is increasing. A number of studies show that the number of privacy breaches is keeping pace with this growth [14], [11], [13], [15]. Very often users' requirements are far from the privacy guarantees offered by social networks which do not meet their expectations. The reasons for that are multifold, ranging from the users' lack of knowledge on the underlying technology to fundamental technical issues of the technology itself.

We are here only concerned with the fact that the privacy settings currently available in social networks are not suitable for capturing the *dynamic* aspect of privacy policies. That is, privacy policies should take into account that the networks *evolve*, as well as the privacy preferences of the users. The privacy policy may "evolve" due to explicit changes done by the users (e.g., a user may change the audience of an intended post to make it more restrictive), or because the privacy policy is dynamic *per se*. Examples of the latter, are for instance: *"My boss cannot know my location between 20:00-23:59 every day"*, or *"Only my friends can see the pictures I am tagged in from Fridays at 20:00 till Mondays at 08:00"*. These are recurrent policies triggered by some time events ("every day between 20:00 and 23:59", and "every week from Friday at 20:00 till Monday at 08:00"). Other policies may be activated or deactivated by certain events: *"Only up to 3 posts, disclosing my location, are allowed per day in my timeline"*.

In this paper we present a formal framework to express evolving privacy policies. We take $\mathcal{FPPF}$ [17] as a point of departure. $\mathcal{FPPF}$ is a formal framework for privacy policies which consists of: i) A generic social network model (SNM); ii) A knowledge-based logic ($\mathcal{KBL}$) to reason about the social network and privacy policies; iii) A formal language ($\mathcal{PPL}$) to describe privacy policies (based on the previous logic). $\mathcal{FPPF}$ is a an expressive privacy policy framework able to represent all privacy policies for social networks like Facebook and Twitter, and beyond [17]. Though rich in what concerns its expressiveness, $\mathcal{FPPF}$ is not suitable to express evolving privacy policies, in the sense discussed above. In summary, our contributions in this paper are:

i) We introduce $\mathcal{TFPPF}$, an extension of $\mathcal{FPPF}$ able to represent recurrent privacy policies parametrised with timed intervals. For that we syntactically extend the privacy policy language $\mathcal{PPL}$ with timed intervals and recurrent behaviour (*timed* $\mathcal{PPL}$), and the underlying epistemic logic $\mathcal{KBL}$ with temporal operators (*box* and *diamond*) and a *learn* operator (*temporal* $\mathcal{KBL}$). Policies in $\mathcal{PPL}$, and formulae in $\mathcal{KBL}$, are interpreted over (finite) timed traces.
ii) We study properties of both the new privacy policy language and the underlying logic, in particular showing that the new operator *learn* cannot be derived. We prove decidability of the satisfaction relation for the logic, and of the conformance relation for the policy language.

The paper is structured as follows, Section II introduces all the elements of the new privacy policy framework. More precisely, in Section II-A we introduce temporal $\mathcal{KBL}$; in Section II-B we describe timed SNMs, we define a satisfaction relation for formulae of the previous logic, we show that the model-checking problem is decidable, and study the properties of the new learning modality; in Section II-C we present the privacy policy language timed $\mathcal{PPL}$, we define what means for a privacy policy to be in conformance to a timed SNM and show that this procedure is decidable. In Section III we analyse the complexity of the $\mathcal{KBL}$ model checking problem, and we provide an alternative optimised model checking algorithm. All the proofs are in the appendix with the exception of Theorem 1 which is presented in Section III.

II. Timed $\mathcal{FPPF}$

In this section, we introduce the extension of $\mathcal{FPPF}$ with time, which contains the following elements:

a) A knowledge-based logic $\mathcal{TKBL}$ with additional temporal modalities, inspired by temporal logics such as LTL.
b) A social network model (as defined for $\mathcal{FPPF}$), together with the notion of traces, i.e., sequences of these models that capture the evolution of an social network which we use to give semantics the previous logic. We use $\mathcal{SN}$ to denote the universe of all possible social network models.
c) A privacy policy language ($\mathcal{TPPL}$), enabling the user to define a (possibly recurring) time window in which their policy should be enforced.

Together, these parts form the new *Timed First-Order Privacy Policy Framework*, $\mathcal{TFPPF}$. In the following sections, we describe each of the components separately.

A. Temporal $\mathcal{KBL}$

$\mathcal{TKBL}$ is a *temporal knowledge-based logic* for social networks. It contains the knowledge modality present in all epistemic logics [9] and the temporal modalities box and diamond. Additionally, it includes: i) Two special types of predicates, *connection* and *action* predicates. Connection predicates represent the "social" connections between users. For instance, *friends*, *colleagues*, *family*, *co-workers*, and so forth. On the other hand, action predicates model the actions which are permitted to be executed by a user. For example, Alice can send a friend request to Bob or Alice can join events created by Bob. We use $\mathcal{C}$ and Σ to denote the set of connection and action predicates, respectively; ii) A modality to represent learning. It will allow us to differentiate between the moment some piece of information has been learnt or whether it is known. We study the relation between the learning and knowledge modalities in the following section.

Let $\mathcal{T}$ be a set (which will refer as *vocabulary*) which consists of *predicate symbols* (p), *function symbols* (f) and *constant symbols* (c). Predicate and function symbols has some implicit arity which corresponds to the number of arguments they take. We assume an infinite supply of variables, which we write as x, y and so on. We can form terms using the elements of $\mathcal{T}$ as follows: $s ::= c \mid x \mid f_i(\vec{s})$ where $\vec{s}$ is a tuple of terms respecting the arity of f_i. Let Ag be a finite set of *agents*. The syntax of $\mathcal{TKBL}$ is defined as follows.

Definition 1 (Syntax of $\mathcal{TKBL}$)**.** *Given agents $i, j \in Ag$, a nonempty set of agents $G \subseteq Ag$, the predicate symbols $a_n(i,j)$, $c_m(i,j)$, $p(\vec{s})$ where $m \in \mathcal{C}$ and $n \in \Sigma$, and a variable x. The* syntax of the timed knowledge-based logic *$\mathcal{TKBL}$ is inductively defined as:*

$$\begin{array}{lcl} \varphi & ::= & \varphi \wedge \varphi \mid \neg\varphi \mid \forall x.\varphi \mid \Box\varphi \mid \Diamond\varphi \mid \xi \\ \xi & ::= & \psi \mid L_i\psi \\ \psi & ::= & \rho \mid \psi \wedge \psi \mid \neg\psi \mid \forall x.\psi \mid K_i\psi \mid D_G\psi \\ \rho & ::= & c_m(i,j) \mid a_n(i,j) \mid p(\vec{s}) \end{array}$$

The remaining epistemic modalities are defined as follows: $S_G\varphi \triangleq \bigvee_{i\in G} K_i\varphi$; $E_G\varphi \triangleq \bigwedge_{i\in G} K_i\varphi$; $SL_G\varphi \triangleq \bigvee_{i\in G} L_i\varphi$; $EL_G\varphi \triangleq \bigwedge_{i\in G} L_i\varphi$. □

Note that in the version of the logic presented here L_i cannot appear within the scope of a K_j operator. We use $\mathcal{F}_{\mathcal{TKBL}}$ to denote the set of all well-formed formulae of $\mathcal{TKBL}$ according to category φ. Formulae in ψ, represent the $\mathcal{KBL}$ logic [16]; $\mathcal{F}_{\mathcal{KBL}}$ denote the set of well-formed $\mathcal{KBL}$ formulae.[1] The epistemic modalities stand for: $K_i\varphi$, agent i knows φ; $L_i\varphi$, agent i learnt φ; $S_G\varphi$, someone in the group G knows φ; $E_G\varphi$, everyone in the group G knows φ; $SL_G\varphi$, someone in the group G learnt φ; $EL_G\varphi$, everyone in the group G learnt φ; $D_G\varphi$, φ is distributed knowledge in the group G. We use the following operators as syntactic sugar for permission: $P_i^j a_n := a_n(i,j)$, agent i is permitted to execute action a_n to agent j; $SP_G^j a_n := \bigvee_{i\in G} a_n(i,j)$, 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 is allowing 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.

B. Semantics of $\mathcal{TKBL}$

$\mathcal{TKBL}$ formulae will be interpreted over traces of social network models. These models are defined as social graphs [8] with agents, their knowledge bases and privacy policies, and a first-order relational structure.

Definition 2 (Social Network Model [16])**.** *Given a set of privacy policies Π, and a finite set of agents $Ag \subseteq \mathcal{AU}$ from a universe $\mathcal{AU}$; a* social network model (SNM) *is a social graph of the form $\langle Ag, \mathcal{A}, KB, \pi\rangle$, where*

- *Ag is a nonempty finite set of* nodes *representing the agents of the social network;*
- *$\mathcal{A}$ is a first-order relational structure over the SNM, consisting of a set of predicate symbols, function symbols and constant symbols interpreted over a domain from a set $\{D_o\}_{o\in\mathcal{D}}$, where $\mathcal{D}$ is a set of indexes for domains;*
- *$KB : Ag \to 2^{\mathcal{F}_{\mathcal{KBL}}}$ is a function returning the set of accumulated knowledge for each agent, stored in what we call the* knowledge base *of the agent. We write KB_i to denote $KB(i)$;*
- *$\pi : Ag \to 2^{\Pi}$ is a function returning the set of privacy policies of each agent. We write π_i for $\pi(i)$.*

[1]For simplicity of presentation we leave out the *common knowledge* operator from the logic. Though rather technical its treatment would only need the machinery from standard epistemic logic.

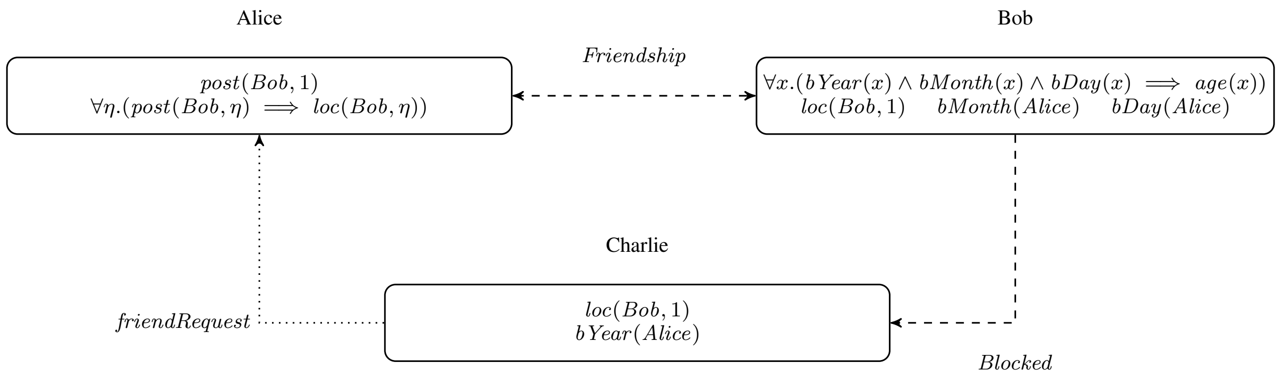
Alice
$post(Bob,1)$
$\forall\eta.(post(Bob,\eta) \implies loc(Bob,\eta))$
Friendship
Bob
$\forall x.(bYear(x) \wedge bMonth(x) \wedge bDay(x) \implies age(x))$
$loc(Bob,1)$ $bMonth(Alice)$ $bDay(Alice)$
Charlie
$loc(Bob,1)$
$bYear(Alice)$
friendRequest
Blocked

Fig. 1. Example of Social Network Model

The knowledge base KB_i of each agent i contains the explicit knowledge that the agent has. Agents not only posses this explicit knowledge, but also anything that can be derived using the **S5** axiomatisation of epistemic logic [9] from the explicitly formulae in their knowledge bases.

Definition 3. *A* derivation *of a formula $\varphi \in \mathcal{F}_{\mathcal{KBL}}$, is a finite sequence of formulae $\varphi_1, \varphi_2, \dots, \varphi_n = \varphi$ where each φ_i, for $1 \leq i \leq n$, is either an instance of the axioms or the conclusion of one of the derivation rules of the* ***S5*** *axiomatisation which premises has already been derived, i.e., it appears as φ_j with $j < i$.*

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

Example 1. *Let SN be an SNM consisting of three agents Alice, Bob and Charlie, $Ag = \{Alice, Bob, Charlie\}$; the friend request action, $\Sigma = \{friendRequest\}$; and the connections Friendship and Blocked, $\mathcal{C} = \{Friendship, Blocked\}$.*

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

*The predicates inside each node represent the agents' knowledge. Charlie and Bob have the predicate $loc(Bob,1)$ inside the node, meaning that both know location number 1 of Bob. Besides predicates, agents' nodes may also contain $\mathcal{KBL}$ formulae that may increase the knowledge of the agents. For instance, Alice knows $loc(Bob,1)$ implicitly. Since the deductive engine (defined by the function Cl) includes the rule social network*Modus Ponens*, Alice can derive that, if she has access to a post of Bob, she can infer his location, i.e. $\forall\eta.post(Bob,\eta) \implies loc(Bob,\eta)$. Alice has access to $post(Bob,1)$, therefore she can infer $loc(Bob,1)$.* □

We use *traces* to capture the evolution of a social network. A trace is a sequence of pairs consisting of a SNM together with a *timestamp*. A timestamp t is a natural number representing the number of milliseconds elapsed since January 1, 1900, 00:00:00.000. We could have chosen a large number of equally good starting points; there is no specific reason for choosing 1900. We will use $\mathbb{T}$ to denote the set of timestamps. We will also use a more human-readable format of `YYYY-MM-DD hh:mm:ss.sss` for individual timestamps, optionally skipping the time part – then we assume it defaults to 00:00:00.000. Based on this, 2016-03-26 10:59:08.234 or 2000-01-01 (which represents 2000-01-01 00:00:00.000) or 1950-12-10 15:35:00.474 are all valid timestamps.

The intuitive meaning of a trace is that each SNM is a snapshot of the social network at point t, as if we froze the network, along with the knowledge and relationships between its agents, at that moment. In addition, we also demand the trace to be finite.

Definition 4 (Trace)**.** *A* trace *σ is a finite sequence $\sigma = \langle (SN_0, t_0), (SN_1, t_1), \dots, (SN_k, t_k) \rangle$ such that, for all $0 \leq i \leq k$, $SN_i \in \mathcal{SN}$ and $t_i \in \mathbb{T}$.*

The (untimed) evolution of a social network was formalised in [16] using a transition relation $SN \xrightarrow{e} SN'$ where SN and SN' are SNMs and e is an event from a set EVT (the set of all the events that can be executed in the social network). Here we extend this transition relation with timestamps, formally $\rightarrow \subseteq \mathcal{SN} \times EVT \times \mathbb{T} \times \mathcal{SN}$. The details of how the elements of the SNM are changed because of the execution of the event are formally described in [16], but they are not relevant for the purpose of this paper. In order to explicitly define which event led to a given SNM in a trace, we sometimes write $SN_0 \xrightarrow{e,t_1} SN_1$ to denote $\langle (SN_0, t_0), (SN_1, t_1) \rangle$ where $SN_0, SN_1 \in \mathcal{SN}$, $e \in EVT$ and $t_0, t_1 \in \mathbb{T}$.

We also introduce the notion of *well-formed trace*, as being a trace satisfying the following two conditions. First, timestamps are strictly ordered from smallest to largest. Second, the transition from (SN_n, t_n) to (SN_{n+1}, t_{n+1}) occurs due to the execution of an event $e \in EVT$ at time t_{n+1}.

Definition 5 (Well-formed traces). *Let $\sigma = \langle (SN_0, t_0), (SN_1, t_1), \ldots, (SN_k, t_k) \rangle$ be a trace. σ is* well-formed *if the following two conditions hold:*

i) Let $n \in \mathbb{N}$. Then for any i, j such that $0 \le i, j \le n$ and $i < j$, it is the case that $t_i < t_j$.

ii) Let $n \in \mathbb{N}$. For all $(SN_n, t_n), (SN_{n+1}, t_{n+1}) \in \sigma$, it is the case that $SN_n \xrightarrow{e, t_{n+1}} SN_{n+1}$ where $e \in EVT$.

We use $\mathcal{WFT}$ to denote the set of all well-formed traces. To make it easier to reason about the attributes of the whole trace, based on the attributes of individual networks, we use the following shortcuts to refer to the SNMs in a trace: $\sigma[t]$ denotes the SNM SN such that $(SN, t) \in \sigma$; given $t_1, t_2 \in \mathbb{T}$ such that $t_1 \le t_2$, $\sigma[t_1..t_2]$ represents the well-formed subtrace of σ starting with the smallest $t \ge t_1$ such that $(SN, t) \in \sigma$, and ending with the largest $t \le t_2$ such that $(SN, t) \in \sigma$.

Often we need to refer to the components of a specific SNM in a trace. For that purpose we use a superscript in the components of SNMs to indicate the SNM of the trace to which the element belongs. For example, given σ and t, $Ag^{\sigma[t]}$ stands for the set of agents in $\sigma[t]$, $D_o^{\sigma[t]}$ for a domain in $\mathcal{A}^{\sigma[t]}$ and $KB_i^{\sigma[t]}$ for i's knowledge base in $\sigma[t]$.

Furthermore, let Ag_σ contain all agents present in the trace, $Ag_\sigma = \bigcup_{(SN,t)\in\sigma} Ag^{SN}$, and $\mathbb{T}_\sigma$ denote the set of all timestamps associated with SNMs in the trace σ: $\mathbb{T}_\sigma = \{t \mid (SN, t) \in \sigma\}$.

Example 2. *Consider a social network with the event checkin, which discloses the location of users to all their friends, and the event opennewsfeed which retrieves all the posts, pictures, locations, etc., that a user has access to. Let $\sigma = \langle (SN, 0), (SN', 1), (SN'', 2) \rangle$ be the following well-formed trace $SN \xrightarrow{checkin(Alice),1} SN' \xrightarrow{opennewsfeed(Bob),2} SN''$. See Fig. 2 for a graphical representation of σ.*

It consists of the agents Alice and Bob, $Ag_\sigma = \{Alice, Bob\}$, but new agents could be added by executing a sign up event. In this trace there are 3 timestamps, $\mathbb{T}_\sigma = \{0, 1, 2\}$. In the initial SNM, SN or $\sigma[0]$, Alice and Bob are friends, $(Alice, Bob) \in C^{\sigma[0]}_{Friendship}$. On the other hand, there is no knowledge neither in Alice's nor Bob's knowledge bases. At time 1 Alice discloses her location by performing a checkin event. Of course, she knows the location she just shared $KB^{\sigma[1]}_{Alice} \vdash loc(Alice, 1)$ (we use 1 as resource id to the location of Alice after she executes the checkin). Bob, however, did not check his newsfeed until time 2. Because of this it is at time 2 when he acquires the knowledge about Alice's location, i.e., $KB^{\sigma[1]}_{Bob} \nvdash loc(Alice, 1)$ and $KB^{\sigma[2]}_{Bob} \vdash loc(Alice, 1)$. □

In what follows we define what means for a formula to be satisfied in a trace.

Definition 6 (Satisfaction). *Given a well-formed trace $\sigma \in \mathcal{WFT}$, agents $i, j \in Ag_\sigma$, a finite set of agents $G \subseteq Ag_\sigma$, formulae $\varphi, \psi \in \mathcal{F}_{\mathcal{TKBL}}$, $m \in \mathcal{C}$, $n \in \Sigma$, $o \in \mathcal{D}$, and $t, t' \in \mathbb{T}_\sigma$, the* satisfaction relation $\models$ *is defined as shown in Table I.*

We use a special agent called *environment* (or simply e) which defines the truth of predicates of the type $p(\vec{s})$. The

$\sigma, t \models \Box\varphi$	iff	for all $t' \in \mathbb{T}_\sigma$, $t' \ge t$, $\sigma, t' \models \varphi$
$\sigma, t \models \Diamond\varphi$	iff	there exists $t' \in \mathbb{T}_\sigma$, $t' \ge t$, such that $\sigma, t' \models \varphi$
$\sigma, t \models \neg\varphi$	iff	$\sigma, t \not\models \varphi$
$\sigma, t \models \varphi \land \psi$	iff	$\sigma, t \models \varphi$ and $\sigma, t \models \psi$
$\sigma, t \models \forall x.\varphi$	iff	for all $v \in D_o^{\sigma[t]}$, $\sigma, t \models \varphi[v/x]$
$\sigma, t \models c_m(i, j)$	iff	$(i, j) \in C_m^{\sigma[t]}$
$\sigma, t \models a_n(i, j)$	iff	$(i, j) \in A_n^{\sigma[t]}$
$\sigma, t \models p(\vec{s})$	iff	$KB_e^{\sigma[t]} \vdash p(\vec{s})$
$\sigma, t \models K_i\varphi$	iff	$KB_i^{\sigma[t]} \vdash \varphi$
$\sigma, t \models L_i\varphi$	iff	$\sigma, t \models K_i\varphi$ and $\nexists t' < t$ such that $\sigma, t' \models K_i\varphi$
$\sigma, t \models D_G\varphi$	iff	$(\bigcup_{i\in G} KB_i^{\sigma[t]}) \vdash \varphi$

TABLE I
SATISFACTION RELATION FOR $\mathcal{TKBL}$

environment's knowledge base (KB_e) contains all predicates which are true in the real world, e.g. $location(Alice) = \text{“}Sweden''$. In Table I, $C_m \subseteq Ag \times Ag$ and $A_n \subseteq Ag \times Ag$ are binary relations used to interpret connection and actions predicates, respectively. We define $E_G^{k+1}\varphi$ as $E_G E_G^k \varphi$, where $E_G^0\varphi$ is equal to φ. We use $\varphi[v/x]$ to denote the usual capture-free substitution in first-order logic. For simplicity, we tacitly assume that each variable v is mapped to its own domain.

In their knowledge bases agents store all the information they get access to. At a given moment in time, the knowledge base of an agent contains all the information the agent have seen so far. Agents do not forget any piece of information since events updating knowledge can only add new formulae, but never remove them. This was formally defined in [16] where we describe the operational semantics rules which capture the evolution of SNMs. Formally, for any σ and $t \ge 1$ it always holds that

$$KB_i^{\sigma[t]} = KB_i^{\sigma[t-1]} \cup \Phi \qquad (1)$$

where $\Phi \in 2^{\mathcal{F}_{\mathcal{KBL}}}$ is a set of formulae representing the new knowledge that i learnt at time t. Given the above, it is easy to show that the latest knowledge base will always have the union of all the formulae that the agent had access to during the execution of the trace. Hence the following trivially follows from (1):

$$\bigcup_{t' \le t} KB_i^{\sigma[t']} = KB_i^{\sigma[t]}.$$

Therefore in order to check whether some piece of information is derivable from the knowledge base at some time t it suffices to check derivability from $KB_i^{\sigma[t]}$. On the other hand, learning is new knowledge (i.e., it was not known before) that is acquired at a given moment in time. As shown in Table I, the difference with knowledge is that for learning we require that the new information cannot be derived in any knowledge base of the trace from a time previous to the one in which the formula is being evaluated. For instance, if Alice posts her location at time 3 and Bob is part of the audience of this post, then Bob has learnt Alice's location at time 3, formally $\sigma, 3 \models L_{Bob} loc(Alice)$. Moreover, it holds that Bob knows Alice's location for any timestamp greater than 3 or $\sigma, t \models K_{Bob} loc(Alice)$ for $t \ge 3$. Note

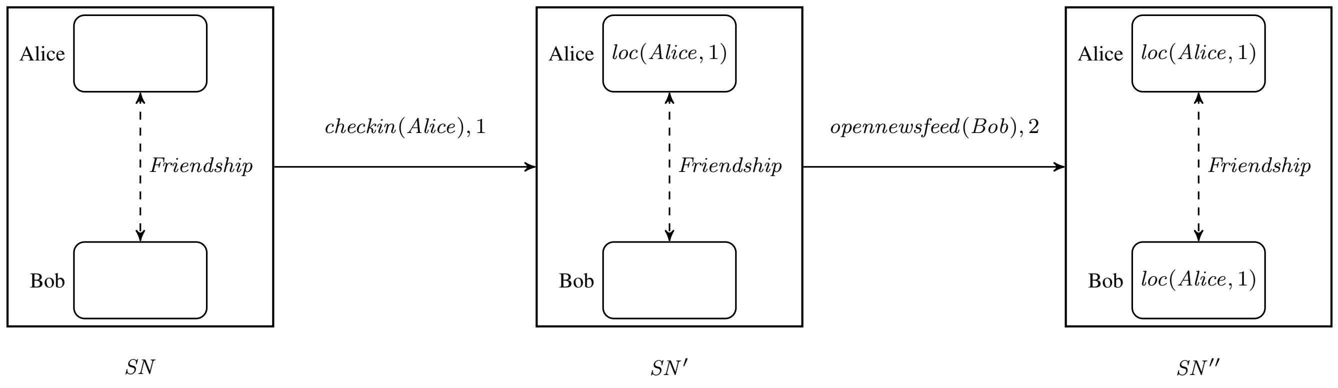

Fig. 2. Example of a Trace of SNMs

that at time 3 Bob learnt Alice's location, but also knows it, $\sigma, 3 \models L_{Bob} loc(Alice) \land K_{Bob} loc(Alice)$.

It always holds that if the agents learn something, then they know it. We call this the *learning axiom*:

$$\text{L. } L_i\varphi \implies K_i\varphi.$$

Lemma 1. *The learning axiom is sound with respect to traces of SNMs.*

It is easy to show that knowing does not imply learning in general. For instance, if $KB_i^{\sigma[t']} \vdash \varphi$ for some $t' < t$ then we have that $\sigma, t \models K_i\varphi$ holds but $\sigma, t \models L_i\varphi$ does not hold.

Another assumption in our models is that of *perfect recall*. When agents in the system learn something, they know it forever. The *perfect recall axiom* is formally defined as

$$\text{PF. } K_i\varphi \implies \Box K_i\varphi.$$

Lemma 2. *The perfect recall axiom is sound with respect to traces of SNMs.*

Example 3. *Let σ be the trace introduced in Example 2. Now we can check whether Bob learns Alice's location after she performs the checkin action:*

$$\begin{aligned}\sigma, 0 \models \Box(&friend(Alice, Bob) \land checkin(Alice) \implies \\ &\exists x. \Diamond L_{Bob} loc(Alice, x))\end{aligned} \tag{2}$$

The $\Box$ operator requires that the implication over it applies for all $t \in \mathbb{T}_\sigma$ such that $t \geq 0$, which in this example are 0, 1 and 2. The $\Diamond$ operator represents that whenever the premise holds, there will be a time in the future where Bob will learn a location of Alice. In order to determine whether it is satisfied over σ, we look at all the elements of the formula. The first conjunct of the premise, $friend(Alice, Bob)$, is already satisfied in SN and it remains true for all the trace, i.e., $(Alice, Bob) \in A_{Friendship}^{\sigma[t]}$ for $t \geq 0$. $checkin(Alice)$ is a predicate representing that Alice executed the event checkin. As mentioned, this general purpose predicates are check in the environment's knowledge base. It first becomes true in SN', i.e., $checkin(Alice) \in KB_e^{\sigma[t]}$ for $t \geq 1$. SN" is the resulting SNM after Bob opens his newsfeed. At this moment he learns Alice's location, i.e., $KB_{Bob}^{\sigma[2]} \vdash loc(Alice, 1)$. Therefore, since the premise holds at time 1 and at time 2 Bob learns the location we conclude that (2) holds. $\Box$

Even though the formula of the previous example hold in σ, it does not guarantee that the location learnt by Bob is the one disclosed during Alice's checkin. This is because the existential quantification ranges over all the location identifiers. Nevertheless, (2) stated that *a location* has been learnt, and indeed, it happened.

The model-checking problem for $\mathcal{KBL}$ formulae in SNMs is decidable [16]. The addition of the new modalities preserves this property for the new $\mathcal{TKBL}$.

Theorem 1 (Model-checking)**.** *Determining whether a formula $\varphi \in \mathcal{F}_{\mathcal{TKBL}}$ is satisfied in a trace $\sigma \in \mathcal{WFT}$ at a given timestamp $t \in \mathbb{T}_\sigma$, that is checking $\sigma, t \models \varphi$, is decidable.*

A proof of the theorem and analysis of its complexity is presented in Section III.

C. Timed $\mathcal{PPL}$

We would like to equip the users of a social network with additional power in defining their privacy policies. This is done by extending the original privacy policy language $\mathcal{PPL}$ with time fields, which enable the user to specify a possibly recurring time window frame in which their policy should be enforced. The basic form of the privacy policies is as follows $[\![\varphi \implies \neg\psi]\!]_{\text{a}}^{[\text{ start } | \text{ duration } | \text{ recurrence }]}$ or $[\![\neg\psi]\!]_{\text{a}}^{[\text{ start } | \text{ duration } | \text{ recurrence }]}$ if the policy has no conditions. The *start* field is mandatory, but *recurrence* and *duration* are optional. Table II shows the intervals $0, 1, \ldots, i$ where a policy must be enforced according to its parameters. If the *recurrence* field is not defined, then a policy should be only enforced in interval 0. We refer to the new language as *timed privacy policy language*, $\mathcal{TPPL}$.

In addition to the notion of a timestamp, we define a related notion of *duration*. A duration d is a positive natural number representing the number of milliseconds elapsed between two points in time. Simply put, it stands for the absolute difference of two timestamps $|t_2 - t_1|$, i.e., the time elapsed

Interval	Start	End
0	start	start + duration
1	start + recurrence	start + recurrence + duration
2	start + 2 recurrence	start + 2 recurrence + duration
3	start + 3 recurrence	start + 3 recurrence + duration
⋮	⋮	⋮
i	start + i recurrence	start + i recurrence + duration

TABLE II
TIME WINDOWS DEFINED BY THE TIME FIELDS OF A POLICY

between t_1 and t_2. We will use a more human-readable format of durations. For instance, $d = 60000$ will be *1 minute*, $d = 2167236000$ will be *25 days, 2 hours and 36 seconds*, and so on.

The starting time of a policy is always a timestamp, which allows us to pinpoint a specific moment in (real) time from which the policy should be enforced. The other two fields are *duration* and *recurrence*. They are written in the duration format. These two fields define an offset from a specific time based on the first field. Table II shows how the three time fields work together to capture certain time windows. A privacy policy with the time fields [2016-16-02 14:00 | 6 hours] is meant to be enforced on February 16, 2016, from 14:00 to 20:00. On the other hand, [2016-01-01 18:00 | 12 hours | 1 day] stands for every night (from 18:00 to 6:00), starting on January 1, 2016. We are now ready to define the general shape of formulae used in the privacy policy language.

Definition 7 (Syntax of $\mathcal{TPPL}$)**.** *Given agents $i, j \in Ag$, a nonempty set $G \subseteq Ag$, a timestamp s, duration d and recurrence r, a variable x, a formula $\varphi \in \mathcal{F}_{\mathcal{KBL}}$, the predicate symbols $c_m(i,j), a_n(i,j), p(\vec{s}) \in \mathcal{A}$ where $m \in \mathcal{C}$ and $n \in \Sigma$, the* syntax of the privacy policy language with time *$\mathcal{TPPL}$ is inductively defined as:*

$$\begin{aligned}
\delta &::= \delta \wedge \delta \mid \forall x.\delta \mid \tau \\
\tau &::= [\![\neg\alpha]\!]_i^{[\,s\,]} \mid [\![\neg\alpha]\!]_i^{[\,s\,|\,d\,]} \mid [\![\neg\alpha]\!]_i^{[\,s\,|\,d\,|\,r\,]} \mid \\
&\quad [\![\varphi \implies \neg\alpha]\!]_i^{[\,s\,]} \mid [\![\varphi \implies \neg\alpha]\!]_i^{[\,s\,|\,d\,]} \mid \\
&\quad [\![\varphi \implies \neg\alpha]\!]_i^{[\,s\,|\,d\,|\,r\,]} \\
\alpha &::= \alpha \wedge \alpha \mid \forall x.\alpha \mid c_m(i,j) \mid a_n(i,j) \mid \alpha\prime \\
\alpha\prime &::= \gamma\prime \mid L_i\gamma\prime \\
\gamma\prime &::= K_i\gamma \mid D_G\gamma \\
\gamma &::= \gamma \wedge \gamma \mid \neg\gamma \mid p(\vec{s}) \mid \gamma' \mid c_m(i,j) \mid a_n(i,j) \mid \forall x.\gamma
\end{aligned}$$

Note that L_i cannot appear under the scope of a knowledge modality. We will denote the set of all formulae of $\mathcal{TPPL}$ as $\mathcal{F}_{\mathcal{TPPL}}$ and the set of all formulae created using the α category (the restrictions) as $\mathcal{F}^{\mathcal{R}}_{\mathcal{TPPL}}$. Now we introduce the notion of *conformance* of a privacy policy in a trace.

Definition 8 (Conformance)**.** *Given a well-formed trace $\sigma \in \mathcal{WFT}$, an agent $i \in Ag_\sigma$, formulae $\varphi \in \mathcal{F}_{\mathcal{TKBL}}, \alpha \in \mathcal{F}^{\mathcal{R}}_{\mathcal{TPPL}}$, $o \in \mathcal{D}$, and $\delta, \delta_1, \delta_2 \in \mathcal{F}_{\mathcal{TPPL}}$, the conformance relation $\models_C$ is defined as shown in Table III.*

$\sigma \models_C \delta_1 \wedge \delta_2$	iff	$\sigma \models_C \delta_1 \wedge \sigma \models_C \delta_2$
$\sigma \models_C \forall x.\delta$	iff	for all $v \in D_o^{\sigma[t]}$, $\sigma \models_C \delta[v/x]$
$\sigma \models_C [\![\neg\alpha]\!]_{\mathrm{i}}^{[\,\mathrm{s}\,\mid\,\mathrm{d}\,\mid\,\mathrm{r}\,]}$	iff	for all $c \in \mathbb{N}_0$ such that $0 \leq s + cr \leq \max(\mathbb{T}_\sigma)$, $\sigma \models_C [\![\neg\alpha]\!]_{\mathrm{i}}^{[\,\mathrm{s+cr}\,\mid\,\mathrm{d}\,]}$
$\sigma \models_C [\![\neg\alpha]\!]_{\mathrm{i}}^{[\,\mathrm{s}\,\mid\,\mathrm{d}\,]}$	iff	$\sigma[s \,..\, s+d], s \models \Box(\neg\alpha)$
$\sigma \models_C [\![\neg\alpha]\!]_{\mathrm{i}}^{[\,\mathrm{s}\,]}$	iff	$\sigma[s \,..\,], s \models \Box(\neg\alpha)$
$\sigma \models_C [\![\varphi \Rightarrow \neg\alpha]\!]_{\mathrm{i}}^{[\,\mathrm{s}\,\mid\,\mathrm{d}\,\mid\,\mathrm{r}\,]}$	iff	for all $c \in \mathbb{N}_0$ such that $0 \leq s + cr \leq \max(\mathbb{T}_\sigma)$, $\sigma \models_C [\![\varphi \Rightarrow \neg\alpha]\!]_{\mathrm{i}}^{[\,\mathrm{s+cr}\,\mid\,\mathrm{d}\,]}$
$\sigma \models_C [\![\varphi \Rightarrow \neg\alpha]\!]_{\mathrm{i}}^{[\,\mathrm{s}\,\mid\,\mathrm{d}\,]}$	iff	$\sigma[s \,..\, s+d], s \models \Box(\varphi \Rightarrow \neg\alpha)$
$\sigma \models_C [\![\varphi \Rightarrow \neg\alpha]\!]_{\mathrm{i}}^{[\,\mathrm{s}\,]}$	iff	$\sigma[s \,..\,], s \models \Box(\varphi \Rightarrow \neg\alpha)$

TABLE III
CONFORMANCE RELATION FOR $\mathcal{TPPL}$

We use $\mathbb{N}_0$ to the denote the set of natural numbers including 0. The relation $\models_C$ is given in terms of the satisfaction relation $\models$. First, the recurrence parameter determines the time windows in which the policy must hold. This is done by getting the starting timestamps of each time window, i.e., from 0 to $\max(\mathbb{T}_\sigma)$. Given that we consider finite traces there will be a finite number of time windows. Afterwards, subtraces from the calculated starting timestamp plus the duration of the policy are checked. In particular, we use *box* to check that all SNMs satisfy the privacy policy. When the duration is not specified we check from the starting timestamp to the end of the trace.

Theorem 2 (Checking conformance)**.** *Determining whether a privacy policy $\varphi \in \mathcal{F}_{\mathcal{TPPL}}$ is in conformance with a trace $\sigma \in \mathcal{WFT}$, that is checking $\sigma \models_C \varphi$, is decidable.*

In what follows we show some examples of using $\mathcal{TPPL}$ to encode evolving privacy policies.

Example 4. *On Friday, April 15, 2016, Alice decides that she wants to keep her private life separate from her life as a graduate student. In a social network with privacy policies using $\mathcal{TPPL}$, she can keep her supervisor Bob from learning her location on weekends by defining the following privacy policy:*

$$\delta = [\![\neg L_{Bob}\,loc(Alice)]\!]_{Alice}^{[\textit{ 2016-04-16 }|\textit{ 2 days }|\textit{ 1 week }]}$$

Given a trace of the social network Alice and Bob use, δ would be checked first in the subtrace from Saturday 16th, 00:00, to Monday 18th, 00:00, then again from Saturday 23rd, 00:00, to the end of Sunday 24th, and so on.

In order for the trace to be in conformance with δ, in each of these subtraces, the $\mathcal{TKBL}$ formula $\Box(\neg L_{Bob}\,loc(Alice))$ needs to be satisfied. Based on the satisfaction relation (cf. Def. 1), this is only the case if the formula $\neg L_{Bob}\,loc(Alice)$ is satisfied at every point of the subtrace. This, in turn, means that it must not be the case that $L_{Bob}\,loc(Alice)$ is satisfied in any of the SNMs of the subtrace.

To determine whether $L_{Bob}\,loc(Alice)$, we check that loc(Alice) can be derived from Bob's knowledge base at time

t which represents the timestamp of the SNM currently being checked. In other words, we have to determine whether Bob learnt (either directly, or by inference) $loc(Alice)$ at point t. As t is a time in one of the time windows δ is defined for (i.e., a weekend sometime after April 16th), Bob having access to this particular piece of knowledge would be a violation of Alice's policy, since it would mean Bob managed to learn Alice's location on a weekend. Note, however, that Bob learning Alice's weekend location at any point not *during a weekend is not considered a violation of the policy. This is due to the fact that the policy is not checked outside the time windows it is defined for.* □

Example 5. *Charlie will start a new one-month job on July 1st, 2016, and he would like to ensure that, during this period, when he is at home only his friends can learn about his posts. He achieves this by using the following policy of $\mathcal{TPPL}$*

$$\delta = \forall x.[\![home(Charlie) \wedge \neg friend(Charlie, x) \implies \neg L_x post(Charlie, text)]\!]^{[\text{ 2016-07-01, 31 days }]}_{Charlie}$$

The predicate $home(Charlie)$ is checked by consulting the environment. Checking whether δ is violated ultimately boils down to checking whether all TSNMs in the trace, starting at 2016-07-01 00:00 and ending at 2016-08-01 00:00, satisfy the whole formula inside the policy. It should be noted, however, that the time when the post was actually posted is irrelevant; what matters is when the users learn about it. So if Charlie is working and her colleague Daniel somehow gains access to her post that was originally posted when she was at home, it is not a violation of either of the policies. □

Example 6. *A few months ago Facebook decided to provide a way for users to get over the end of a relationship in an easier way. It is carried out by limiting the information that shows up from the former partner [3]. For instance, pictures, videos or posts from the ex-partner do not appear in the newsfeed. This can be seen as a special set of privacy policies that apply when users breakup. Of course, they can be modelled using $\mathcal{TPPL}$. Consider the privacy policy "if we break up, then you can no longer learn about pictures I am tagged in". Let us say it is Frank who wants to enforce this and Eve is his current girlfriend. He is free to write the following in $\mathcal{TPPL}$:*

$$\delta = \forall \eta.[\![brokenup(Frank, Eve) \wedge taggedin(Frank, \eta) \implies \neg L_{Eve} picture(\eta)]\!]^{[\ t\ ]}_{Frank}$$

Here, too, checking whether δ is violated with respect to a trace means checking the contents of the policy in all SNMs in the trace, starting at time t. So if Eve gains access to a picture Frank is tagged in that is new to her (no matter when it was originally posted) when they are no longer together, Frank's policy will be violated. □

III. $\mathcal{TKBL}$ MODEL-CHECKING

In order to show the proof of Theorem 1, that is, the decidability of the model-checking problem for $\mathcal{TKBL}$, we present a naive model-checking algorithm which implements directly the semantics of $\mathcal{TKBL}$ in Table I.

Lemma 3. *Let σ be a well-formed trace of length n, $t \in \mathbb{T}_\sigma$ be a timestamp, and $\varphi \in \mathcal{F}_{\mathcal{TKBL}}$ be a formula. Let q be maximum number of nested quantifiers in φ and d an upper-bound on the size of the domains for the quantifiers. There is an algorithm that determines, using $O(n \times |\varphi| \times d^q \times |Ag|)$ queries to the epistemic reasoning engine, whether $\sigma, t \models \varphi$.*

Proof. We first expand the universal quantifiers in φ by inductively transforming each subformula $\forall x.\varphi'$ into a conjunction with one conjunct $\varphi'[v/x]$ for each element v in the domain D. The resulting formula is quantifier free and has size $O(|\varphi| \times d^q)$ where d is a bound on the size of the domain and q is the maximum nested stack of quantifiers. Let $\varphi_1, \ldots, \varphi_m$ be the subformulas of the resulting formula, ordered respecting the subformula relation. An easy induction on $k < m$ shows that we can label—for every agent and at every step of the trace—, starting from the earliest time-stamp with either φ_k or $\neg\varphi_k$. We begin with the atomic part, ρ in Def. 1:

- Checking $c_m(e, f)$ and $a_n(e, f)$ can be performed in constant time, simply by checking the model at the given instant, for every agent.
- Checking $p(\vec{s})$ at a given instant t requires one query to the epistemic reasoning engine for $KB_i^{\sigma[t]}$ (for agent i and time stamp t).

Then, for the epistemic part (ψ in Def. 1) we first resolve all operators except L_i:

- Checking $\psi_k = \neg\psi_j$ and $\psi_k = \psi_j \wedge \psi_i$ can be done in constant time for each instant t and agent i, using the induction hypothesis.
- Checking $K_i\psi_j$ requires one query to the epistemic engine for $KB_i^{\sigma[t]} \vdash \psi_j$ per instant.
- Checking $D_G\psi$ at a given instant can be done with $|Ag|$ queries for each instant.

The operator $L_i\psi$ is handled directly by checking the values of $K_i\psi$ in the present instant and t the previous instant t', which has been precomputed as whether the formula ψ is present or not in the label for agent i at instants t and t'.

Finally, for the temporal part (φ in Def. 1), we traverse the trace from the end to the beginning.

- Checking $\Box\varphi$ at instant t requires obtaining φ at t and $\Box\varphi$ at the next instant, using the temporal expansion $\Box\varphi \equiv \varphi \wedge \mathsf{X}\varphi$.
- Checking $\Diamond\varphi$ requires the same information, and hence can also be done in time linear in the lenght of the trace, using the temporal expansion $\Diamond\varphi \equiv \varphi \vee \mathsf{X}\varphi$.

It is easy to see that the semantics of $\mathcal{TKBL}$ is captured by the algorithm and the bounds of the theorem in the number of queries to the epistemic engine are met. □

Computing whether a formula φ can be derived from a collection of knowledge facts is a PSPACE-complete problem by Fagin *et al.* in [9], so reducing the number of queries is essential. Note that in the algorithm presented in Lemma 3 the number of tests to the epistemic engine to resolve a $L_i\varphi$ formula is reduced to one based on the fact that the knowledge of the agents grows monotonically and provided that (a) $K_i\varphi$

is memoized at every step, and (b) the sequence is visited in increasing order.

A. Timestamping knowledge

An alternative solution to the algorithm in Lemma 3 is to add timestamps to formulae in the agents' knowledge bases. If Alice learns Bob's location at time 3, we say that $(loc(Bob), 3) \in KB_{Alice}$. It also allows us to differentiate between a formula that has been learnt twice. For example, Alice can also learn Bob's location at time 7, thus $(loc(Bob), 3), (loc(Bob), 7) \in KB_{Alice}$.

This approach allows us to incrementally remember when facts are learnt by the epistemic engine, but requires to know upfront the formula to model-check. This approach requires to update the notion of derivability (cf. Def. 3) to support timestamps. For simplicity, this time we define a *timed closure* function which computes all derivable formulae from an agent's knowledge base taking into account timestamps. We denote this closure function $Tcl : 2^{\mathcal{F}_{\mathcal{KBL}} \times \mathbb{T}} \rightarrow 2^{\mathcal{F}_{\mathcal{KBL}} \times \mathbb{T}}$, for the formal definition see Appendix Def. 9. Thus the semantics of K_i and L_i would be modified as follows

$$\begin{array}{lll} \sigma, t \models K_i\varphi & \text{iff} & (\varphi, t') \in Tcl(KB_i^{\sigma[t]}) \text{ where } t' \in \mathbb{T} \\ \sigma, t \models L_i\varphi & \text{iff} & (\varphi, t) \in Tcl(KB_i^{\sigma[t]}) \end{array}$$

The timestamps of the formulae that are added to the agents' knowledge base must grow monotonically. In the axioms and derivation rules of $Tcl(KB_i)$ (cf. Def. 9) when two different timestamps are involved in the derivation we always take the maximum for the derived formula. If one timestamp is involved we keep the value. It turns out to be enough to preserve the monotonicity of the derivations.

Lemma 4. *Time in $Tcl(KB_i)$ is monotonic.*

In order to make sure of the correctness of this modification we also require that $Tcl(KB_i)$ preserves the amount of knowledge that the agents can infer with respect to the untimed version $\vdash$.

Lemma 5. *For all $\varphi \in \mathcal{F}_{\mathcal{KBL}}$, $KB_i \vdash \varphi$ iff $(\varphi, t) \in Tcl(KB_i)$ where $t \in \mathbb{T}$.*

As expected adding timestamps to derivations keeps the problem decidable.

Lemma 6. *For all $\varphi \in \mathcal{F}_{\mathcal{KBL}}$ and $t \in \mathbb{T}$, determining whether $(\varphi, t) \in Tcl(KB_i)$ is decidable.*

B. Further Optimizations

There are other optimisations that could be applied to obtain a more practical algorithm. The first observation is that the size d of the domains in a practical SNM can be very large, but the potentially interesing instantiations of a universal quantifier in a formula can typically be bound with a much smaller value. This is the case when the sub-formula within the quantifier is guarded by a limiting predicate, that is the antecedent A of the formula $\forall x.[A \Rightarrow B]$ is, for example, $friend(Charlie, x)$. The only potential instantiations that make the formula true are those friends of $Charlie$. This set can be scoped much better than the whole population of the social network. Another future optmization is to leverage the proof tree of one derivation for a subsequent derivation. Consider the check at time t for $KB_i^{\sigma[t]} \vdash \varphi \wedge \psi$. In a later check $KB_i^{\sigma[t+1]} \vdash \psi$ we could leverage the proof at time t to instantaneusly determine that ψ is derivable. Of course, this approach requires to always keep the proof tree in memory, which might be problematic if the knowledge bases of the agents grow quickly.

IV. Related Work

To the best of our knowledge, this work is the first attempt to formalise privacy policies for social networks which depend on time. However, specifying and reasoning about temporal properties in multi-agent systems using epistemic logic have been subject of study for a long time. It began with the so called *interpreted systems* (IS). In [9] Fagin *et al.* introduce IS as a model to interpret epistemic formulae with temporal operators such as box and diamond. IS have been used for security analyses of multi-agent systems. For instance, IS have been used to formalise the notion of secrecy [10] and information-flow properties such as non-interference [6]. $\mathcal{TKBL}$ has similar semantics to IS, but with the difference that in IS perfect recall is not always assumed. It means that forgetful agents can be modelled, unlike in SNMs.

Recent research has been carried out in extending IS to be able to reason about past or future knowledge. In [7] Moses *et al.* extend K_i with a timestamp $K_{i,t}$. It makes possible to express properties such as "Alice knows at time 5 that Bob knew p at time 3", i.e., $K_{Alice,5}K_{Bob,3}\ p$. $\mathcal{TKBL}$ is not as expressive as that of Moses, since formulae in $\mathcal{TKBL}$ cannot talk about future of past knowledge. In $\mathcal{TKBL}$ we can only differentiate between learning and knowing. Nevertheless, in the knowledge bases proposed in Section III we include timestamps indicating when the agents learnt the information. Thus, we claim that formulae in Moses' language could be interpreted in those knowledge bases. Using an approach similar to Moses' epistemic logic would make it possible to define learning in terms of knowledge, since the language permits to syntactically specify the timestamp of the knowledge.

In [18] Woźna & Lomuscio present TCTLKD a combination of epistemic logic, CTL, a deontic modality and real time. Formulae in TCTLKD are more expressive than both of the languages we present in this paper $\mathcal{TKBL}$ and $\mathcal{TPPL}$. The models used to interpret formulae in TCTLKD are also more complex than the ones presented in this paper, since they are based on a semantics for a branching logic. They are a combination of timed automata and IS plus a an equivalence relation for modelling permission. For our purpose we do not need such a complex modelling power. Even though it has not been formally studied, we claim that the complexity of the model-checking problem of formulae in TCTLKD is much higher than that of $\mathcal{TKBL}$.

V. Final Discussion

We have presented a novel privacy policy framework with support for dynamic recurrent privacy policies that depend on time. It has been done by extending $\mathcal{FPPF}$ [16], [17]. Concretely, we have extended the knowledge-based logic with the temporal modalities $\Box$, $\Diamond$, and the learning modality L_i resulting in $\mathcal{TKBL}$. A satisfaction relation to check formulae in $\mathcal{TKBL}$ has been defined as well. These elements correspond the intermediate tools to the enforcement of privacy policies that depend on time. Additionally, we have studied some properties regarding the relation between knowledge and learning. We have provided the language $\mathcal{TPPL}$ to express timed privacy policies and a conformance relation to check that the policies are not violated during the execution of a trace. Finally, we have proved that checking conformance of a privacy policy in $\mathcal{TPPL}$ and model-checking of $\mathcal{TKBL}$ formulae are decidable.

Yet there are some limitations in $\mathcal{TFPPF}$. Firstly, in $\mathcal{TKBL}$ we cannot write formulae that describe knowledge at a given moment in time. This could be solved by introducing a knowledge modality which includes the timestamp of the knowledge we are interested in checking. It would make the learning modality derivable from knowledge. Secondly, we only check the privacy policies during the time window specified in the policy. This might not be as expressive as one might wish. Consider that Alice enables the following policy "Only my friends can know my pictures during the weekend" (P1). Let Bob be a friend of Alice. If Alice shares a picture on Saturday, Bob will have access to it. Imagine now that on Monday Alice unfriends Bob. At this moment P1 should be violated, because Alice is not a friend with Bob and Bob knows a picture of Alice during the weekend. To enforce this stronger privacy policy we need to extend the logic not only with timestamps in the modalities, but also in the predicates. It will also enable the possibility of having more precise models where the timestamps of predicates represent their own timestamp instead of the time when an agent learnt them.

There exists a prototype implementation of some of the policies of $\mathcal{FPPF}$ in the social network Diaspora* [1], [4]. We are currently investigating how to adapt our implementation to support $\mathcal{TFPPF}$ privacy policies.

References

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

[2] Facebook. https://www.facebook.com/. Accessed: 2016-04-29.

[3] Improving the experience when relationships end*. facebook's newsroom. http://newsroom.fb.com/news/2015/11/improving-the-experience-when-relationships-end/. Accessed: 2016-06-27.

[4] $\mathcal{PPF}$ diaspora*. https://github.com/raulpardo/ppf-diaspora. Accessed: 2016-04-29.

[5] Twitter, inc. https://twitter.com/. Accessed: 2016-04-29.

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

[7] I. Ben-Zvi and Y. Moses. Agent-time epistemics and coordination. In *Logic and Its Applications*, volume 7750 of *LNCS*, pages 97–108. Springer, 2013.

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

[9] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning about knowledge*, volume 4. MIT press Cambridge, 1995.

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

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

[12] A. Lenhart, K. Purcell, A. Smith, and K. Zickuhr. Social media & mobile internet use among teens and young adults. millennials. *Pew Internet & American Life Project*, 2010.

[13] Y. Liu, K. P. Gummadi, B. Krishnamurthy, and A. Mislove. Analyzing facebook privacy settings: User expectations vs. reality. IMC '11, pages 61–70. ACM, 2011.

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

[15] M. Madejski, M. L. Johnson, and S. M. Bellovin. The failure of online social network privacy settings. 2011.

[16] R. Pardo. *Formalising Privacy Policies for Social Networks*. Department of Computer Science and Engineering, Chalmers University of Technology, 2015. Pages 102. Licentiate thesis.

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

[18] B. Woźna and A. Lomuscio. A logic for knowledge, correctness, and real time. In *Computational Logic in Multi-Agent Systems*, volume 3487 of *LNCS*, pages 1–15. Springer, 2004.

Appendix

A. Formal definitions of timed knowledge bases

Definition 9 (Timed Closure of a Knowledge base)**.** *Given the knowledge base of an agent i, KB_i, $Tcl(KB_i)$ satisfies the following properties:*

a) For all $\varphi \in \mathcal{F}_{\mathcal{KBL}}$ and $t \in \mathbb{T}$, If $(\varphi, t) \in Tcl(KB_i)$ then $(\neg\varphi, t) \notin Tcl(KB_i)$,

b) Introduction and elimination rules for conjunction:

$\wedge_I$ - If $(\varphi, t) \in Tcl(KB_i)$ and $(\psi, t') \in Tcl(KB_i)$, then $(\varphi \wedge \psi, \max(t, t')) \in Tcl(KB_i)$

$\wedge_{E_1}$ - If $(\varphi \wedge \psi, t) \in KB_i$, then $(\varphi, t) \in Tcl(KB_i)$,

$\wedge_{E_2}$ - Analogous to $\wedge_{E_1}$ but for ψ,

c) If $(\varphi, t) \in Tcl(KB_i)$ and $(\varphi \implies \psi, t') \in Tcl(KB_i)$, then $(\psi, \max(t, t')) \in Tcl(KB_i)$,

d) If $(\varphi, t) \in Tcl(KB_i)$ then $(K_i\varphi, t) \in Tcl(KB_i)$,

e) If φ is provable in the axiomatisation **S5** *([9]) from $Tcl(KB_i)$, then $\varphi \in Tcl(KB_i)$. Formally:*

A1 - If φ is an instance of a first-order tautology, then $(\varphi, t^\top) \in Tcl(KB_i)$,

A2 - If $(K_i\varphi, t) \in Tcl(KB_i)$ and $(K_i(\varphi \implies \psi), t') \in Tcl(KB_i)$, then $(K_i\psi, \max(t, t')) \in Tcl(KB_i)$,

A3 - If $(K_i\varphi, t) \in Tcl(KB_i)$, then $(\varphi, t) \in Tcl(KB_i)$,

A4 - If $(K_i\varphi, t) \in Tcl(KB_i)$, then $(K_iK_i\varphi, t) \in Tcl(KB_i)$,

A5 - If $(\varphi, t) \notin Tcl(KB_i)$, then $(\neg K_i\varphi, t) \in Tcl(KB_i)$,

R1 - Modus ponens, *it is defined as c),*

R2 - If (φ, t) is provable from no assumptions (i.e., φ is a tautology) then $(K_i\varphi, t) \in Tcl(KB_i)$,

C1 - $(E_G\varphi, t) \in Tcl(KB_i)$ iff $(\bigwedge_{i \in G} K_i\varphi, t) \in Tcl(KB_i)$,

C2 - $(C_G\varphi, t) \in Tcl(KB_i)$ iff $(E_G(\varphi \wedge C_G\varphi), t) \in Tcl(KB_i)$, *RC1 - If* $(\varphi \implies E_G(\psi \wedge \varphi), t)$ *is provable from no assumptions, then* $(\varphi \implies C_G\psi, t) \in \mathit{Tcl}(KB_i)$*,*

D1 - $(D_{\{i\}}\varphi, t) \in \mathit{Tcl}(KB_i)$ *iff* $(K_i\varphi, t) \in \mathit{Tcl}(KB_i)$*,*

D2 - If $(D_G\varphi, t) \in \mathit{Tcl}(KB_i)$*, then* $(D_{G'}\varphi, t) \in \mathit{Tcl}(KB_i)$ *if* $G \subseteq G'$*,*

DA2-DA5 Properties A2, A3, A4 and A5, replacing the modality K_i *with the modality* D_G *for each axiom.*

Remark 1. *The rules* $\wedge_{E_1}$ *and* $\wedge_{E_2}$ *can only be applied if the formula* $\varphi \wedge \psi$ *was explicitly added to the knowledge base to avoid illegal updates of timestamps. If* $\wedge_I$ *was used to construct* $\varphi \wedge \psi$*, the following derivation would be possible. Given that* $(\varphi, 1) \in \mathit{Cl}(KB_i)$ *and* $(\psi, 4) \in \mathit{Cl}(KB_i)$*, by* $\wedge_I$ *we get* $(\varphi \wedge \psi, 4) \in \mathit{Tcl}(KB_i)$*. Now if we apply* $\wedge_{E_1}$ *we can derive* $(\varphi, 4) \in \mathit{Tcl}(KB_i)$*, which adds an updated copy of* φ *to the knowledge base of the agents. This update must be forbidden because it corresponds to unrealistic knowledge.*

Remark 2. *Tautologies in* $\mathit{Tcl}(KB_i)$ *have the special timestamp* $t^\top$*. It is used to represent any timestamp, i.e.,* $t = t^\top$ *for all* $t \in \mathbb{T}$*. When tautologies are used in a derivation involving other premises we will always take the timestamp of the premise that is not a tautology, it is formally guaranteed by defining* $\max(t, t^\top) = t$*.*

B. Proofs

Lemma 1. *The axiom L is sound w.r.t. traces of SNMs.*

Proof. It trivially follows from $\models$ (cf. Def. 6). We show that for all $\sigma \in \mathcal{WFT}$, $t \in \mathbb{T}$ and $\varphi \in \mathcal{F_{KBL}}$ the following holds $\sigma, t \models L_i\varphi \implies K_i\varphi$. Assume that $\sigma, t \models L_i\varphi$. By $\models$, it follows that $KB_i^{\sigma[t]} \vdash \varphi$. Therefore, by $\models$ we can conclude that $\sigma, t \models K_i\varphi$. □

Lemma 2. *The perfect recall axiom is sound w.r.t. traces of SNMs.*

Proof. It trivially follows from $\models$ (cf. Def. 6). We show that for all $\sigma \in \mathcal{WFT}$, $t \in \mathbb{T}$ and $\varphi \in \mathcal{F_{KBL}}$ the following holds $\sigma, t \models K_i\varphi \implies \Box K_i\varphi$. Assume that $\sigma, t \models K_i\varphi$. By $\models$, it follows that $KB_i^{\sigma[t']} \vdash \varphi$ such that $t' \in \mathbb{T}_\sigma$ and $t' \leq t$. By $\models$, $\sigma, t \models \Box K_i\varphi$ holds iff for all $t'' \in \mathbb{T}$ such that $t'' \geq t$. Since at t there exists a t' such that $t' \leq t$ and information cannot disappear from KB_i, for all $t'' \in \mathbb{T}_\sigma$ such that $t'' \geq t$, it holds $KB_i^{\sigma[t'']} \vdash \varphi$, therefore $\sigma, t \models \Box K_i\varphi$ holds. □

Theorem 2. *Determining whether a privacy policy* $\varphi \in \mathcal{F_{TPPL}}$ *is in conformance with a trace* $\sigma \in \mathcal{WFT}$*, that is checking* $\sigma \models_C \varphi$*, is decidable.*

Proof. It follows from the Def. 8. We show it by induction on the structure of the privacy policies. The base cases are $[\![\neg\alpha]\!]_i$ and $[\![\varphi \implies \neg\alpha]\!]_i$ with their corresponding time frames [s] and [s | d]. Since satisfaction of $\mathcal{TKBL}$ formulae is decidable (cf. Theorem 1), we need to establish decidability of computing the time frame. For [s] it is decidable since we consider finite traces and it reduces to checking satisfaction at time s of all the SNMs included in the finite subtrace from s to the end of the original trace. The case [s | d] is analogous to the previous one, but taking a subtrace of the original trace, which length is determined by the duration parameter. The inductive steps are [s | d | r], $\wedge$ and $\forall$. the case The case [s | d | r] splits the trace in time frames according to the recurrence parameter. Given that σ is finite, there is only a finite number of integers from 0 to $\max(\mathbb{T}_\sigma)$, hence the splitting is finite. Moreover, it reduces to the case [s | d], which by induction hypothesis is decidable. As in the proof of Theorem 1 the inductive steps for $\forall$ and $\wedge$ follow from their induction hypothesis. □

Lemma 3. *Time in timed derivations is monotonic.*

Proof. It follows from the definition of $\mathit{Tcl}(KB_i)$ (cf. Def. 9). We show that none of the properties in Def. 9 generate a formula with a timestamp less than that of any of the premises applied in the derivation. Properties a), $\wedge_{E_1}$, $\wedge_{E_2}$, d), A1, A3, A4, A5, R2, C1, C2, RC1, D1, D2, and DA3-DA5 do not change the timestamp of the premise they are applied on. Therefore the timestamp of a formula derived using any of the previous properties will be equal to that of the premise(s). The remaining properties $\wedge_I$, c), A2, R1 and DA2 always take the maximum timestamp of the premises used to derive the conclusion, hence the timestamp of a derived formula will always be greater of equal to that of the premise(s) used in the derivation. Finally, we conclude that none of the properties in $\mathit{Tcl}(KB_i)$ produces a formula with a timestamp smaller to that of the premise(s) used in the derivation, hence time in these derivations is monotonic. □

Lemma 4. *For all* $\varphi \in \mathcal{F_{KBL}}$*,* $\varphi \in \mathit{Cl}(KB_i)$ *iff* $(\varphi, t) \in \mathit{Tcl}(KB_i)$ *where* $t \in \mathbb{T}$*.*

Proof. It trivially follows from the definition of $\mathit{Cl}(KB_i)$ [16]. Given that Tcl (cf. Def. 9) is characterised by the exact same properties and since the operations regarding timestamps (i.e., take the maximum of two timestamps or keep the same timestamp) do not affect the derivations using the axioms and rules of Tcl, we conclude that both sets contain the same formulae. □

Lemma 5. *For all* $\varphi \in \mathcal{F_{KBL}}$ *and* $t \in \mathbb{T}$*, determining whether* $(\varphi, t) \in \mathit{Tcl}(KB_i)$ *is decidable.*

Proof. Checking whether a formula $\varphi \in \mathcal{F_{KBL}}$ is derivable under the axiomatisation **S5** is decidable [9]. By definition, the closure function Cl (and consequently Tcl) are **S5** maximal-consistent sets, hence checking whether $\varphi \in \mathit{Cl}(KB_i)$ is decidable. Therefore, the remaining question is whether the derivation algorithm for timestamps is decidable. For properties a), $\wedge_{E_1}$, $\wedge_{E_2}$, d), A3, A4, A5, R2, C1, C2, RC1, D1, D2, and DA3-DA5, it is trivially decidable since t is not updated. For $\wedge_I$, c), A2, R1 and DA2, it is also decidable since computing the maximum of two numbers is decidable. Finally, when φ is a tautology $(\varphi, \top) \in \mathit{Tcl}(KB_i)$, hence once it is established that φ is a tautology the value of t is irrelevant (since $t = \top$ for all $t \in \mathbb{T}$), and as mentioned, determining whether φ a tautology is decidable. Finally, we conclude that $(\varphi, t) \in \mathit{Tcl}(KB_i)$ is decidable. □