Timed Epistemic Knowledge Bases for Social Networks ⋆

Raúl Pardo[1], César Sánchez[2], and Gerardo Schneider[1]

[1] Department of Computer Science and Engineering,
Chalmers | University of Gothenburg, Sweden.
[2] IMDEA Software Institute, Madrid, Spain.
pardo@chalmers.se, cesar.sanchez@imdea.org, gersch@chalmers.se

Abstract. We present an epistemic logic equipped with time-stamps in atoms and epistemic operators, which enables reasoning about the moments at which events happen and knowledge is acquired or deduced. Our logic includes both an epistemic operator K and a belief operator B, to capture the disclosure of inaccurate information. Our main motivation is to describe rich privacy policies in online social networks (OSNs). Most of today's privacy policy mechanisms in existing OSNs allow only *static policies*. In our logic it is possible to express rich *dynamic policies* in terms of the knowledge available to the different users and the precise time of actions and deductions. Our framework can be instantiated for different OSNs by specifying the effect of the actions in the evolution of the social network and in the knowledge disclosed to each user. We present an algorithm for deducing knowledge and propagating beliefs, which can also be instantiated with different variants of how the epistemic information is preserved through time. Policies are modelled as formulae in the logic, which are interpreted over timed traces. Finally, we show that the model checking problem for this logic, and in consequence policy conformance, is decidable.

1 Introduction

Online Social Networks (OSNs) like Facebook, Twitter and Snapchat have exploded in popularity in recent years. According to a recent survey [1] nearly 70% of the Internet users are active on social networks. Some concerns, including privacy, have arisen alongside this staggering increase in usage. Even though several studies [2–5] report that privacy breaches are growing in number, the most popular OSNs do not provide effective mechanisms to express privacy policies; virtually all privacy policies are static and cannot express timing preferences, such as referring to points in time or how policies evolve.

In [6] we presented a framework to express a limited version of dynamic privacy policies, based on an *epistemic logic* to characterise what users know. Formulae are interpreted over *social network models* which faithfully represent the social graph of

⋆ This research has been partially supported by: the Swedish funding agency SSF under the grant *Data Driven Secure Business Intelligence*, the Swedish Research Council (*Vetenskapsrådet*) under grant Nr. 2015-04154 (*PolUser: Rich User-Controlled Privacy Policies*), the EU H2020 project Elastest (num. 731535), by the Spanish MINECO Project "RISCO (TIN2015-71819-P)" and by the EU ICT COST Action IC1402 ARVI (*Runtime Verification beyond Monitoring*).

OSNs. The policy language in [6] allows to describe, for example, the following policy "*During the weekend only my friends can see my pictures*". However, the previous policy simply activates the static policy "*only my friends can see my pictures*" during weekends. Two restrictions of the logic in [6] are the lack of explicit time, and that only a knowledge modality is available, thus implicitly assuming that the information that users are told is true. This assumption is not realistic in social networks as users may also receive and disclose information that is false or inaccurate, which has raised a growing interest in the detection of fake news [7–9]. To address these issues we introduce here a logic that: (1) is tailored for social networks and allows to express properties based on the social connections between users; (2) combines knowledge and belief to differentiate true knowledge and information that may be false; (3) has time-stamps in modalities and atoms, which allows to refer to the timing of events and when information is learnt. Based on this logic, we introduce a novel privacy policy language that addresses the limitations mentioned above.

Some existing logics include these elements separately. For example, [10] includes time-stamps in atoms and in a belief modality, but it lacks a knowledge modality, and it is not suitable for OSNs (their aim was to reason about AGM belief revision). The logic proposed in [11] can reason about how beliefs spread in Twitter, but it does not include time-stamps. Finally, [12] proposes an axiomatisation of an epistemic logic with knowledge and belief, but without time-stamps in modalities or atoms. Section 4 includes a more detailed comparison of related work.

Contributions: In this paper we introduce a novel logic that combines knowledge, belief and time (Section 2), tailored to define *dynamic privacy policies* (Section 3). More concretely: (1) We extend [6] by equipping atoms and epistemic operators with time instants which allows to derive *learning* and *forget* operators. (2) We equip the logic with *belief* operators, with the restriction that agents cannot believe in something that they know is false, and we define a belief propagation algorithm which guarantees that agents' beliefs are always consistent. This allows us to model OSNs that permit gossiping in which potentially false information can be spread. Analogously, we derive the *accept* and *reject* operators which capture the moment in which an agent starts or stops believing in something. (3) We introduce the notion of *extended knowledge bases* (EKBs), which allow to answer queries of temporal epistemic formulas against the knowledge acquired during a sequence of events, via *timed derivations* labelled with time-windows. This idea allows to instantiate our framework for different OSNs, for example those with eternal memory like Facebook and for ephemeral ones like Snapchat. (4) We prove that the model checking problem for this logic is decidable by providing a model checking algorithm that is also used to check policy conformance. As a result, policy conformance is also decidable. The purpose of this paper is to present an expressive foundation for developing algorithms for detecting privacy violations and for privacy enforcement. We leave the discussion of specialized efficient algorihtms for future work.

2 A Timed Knowledge Based Logic

We introduce here $\mathcal{KBL}_{\mathcal{RT}}$ a knowledge-based first-order logic that includes time-stamped knowledge and belief modalities, and quantification over time-stamps.

2.1 Syntax

Let $\mathcal{T}$ be a *vocabulary* consisting of a set of predicate and function symbols, with some implicit arity, and constant symbols. We assume an infinite supply of variables $x, y, \ldots$ Terms can be built as $s ::= c \mid x \mid f(\vec{s})$ where $\vec{s}$ is a tuple of terms respecting the arity of f. Let $\mathbb{T}$ denote a set of *time-stamps*, which is required to be a non-Zeno totally ordered set, i.e., there is a finite number of instants between any two given instants. We use time-stamps to mark pieces of information or to query the knowledge of the agents at specific instants. Consider also a set of agents Ag, a set of domains $\mathcal{D}$, and a set of events EVT (e.g., share a post or upload a picture). Similarly, we use $\mathcal{C}$ and Σ to denote special sets of predicate symbols that denote connections (between agents) and permissions.

Definition 1 (Syntax of $\mathcal{KBL}_{\mathcal{RT}}$). *Given agents $i, j \in Ag$ a time-stamp $t \in \mathbb{T}$, an event $e \in EVT$, a variable x, a domain $D \in \mathcal{D}$, predicate symbols $c^t(i,j), a^t(i,j), p^t(\vec{s})$ where $c \in \mathcal{C}$ and $a \in \Sigma$, the* syntax of the real-time knowledge-based logic $\mathcal{KBL}_{\mathcal{RT}}$ *is inductively defined as:*

$$\varphi ::= \rho \mid \varphi \wedge \varphi \mid \neg\varphi \mid \forall t \cdot \varphi \mid \forall x : D \cdot \varphi \mid K_i^t\varphi \mid B_i^t\varphi$$
$$\rho ::= c^t(i,j) \mid a^t(i,j) \mid p^t(\vec{s}) \mid \mathit{occurred}^t(e)$$

The epistemic modalities stand for: $K_i^t\varphi$, agent i knows φ at time t; $B_i^t\varphi$, agent i believes φ at time t. We use the following notation as syntactic sugar $P_i^j a^t \triangleq a^t(i,j)$, meaning that "*agent i is permitted to execute action a to agent j at time t*". For example, $P_{Bob}^{Alice}\mathit{friendRequest}^5$ means that Bob is allowed to send a friend request to Alice at time 5. We use $\mathcal{F}_{\mathcal{KBL}_{\mathcal{RT}}}$ to denote the set of all well-formed $\mathcal{KBL}_{\mathcal{RT}}$ formulae. Our logic introduces the following novel notions that have not been considered in other formal privacy policies languages such as [13–15, 6].

- *Time-stamped Predicates*. Time-stamps are explicit in each predicate, including connections and actions. For instance, if Alice and Bob were friends in a certain time period, then the predicate $\mathit{friend}^t(Alice, Bob)$ is true for all t falling into the period, and false for all t outside. This can be seen as the *valid time* in temporal databases [16].
- *Separating Knowledge and Belief.* Not all the information that users see in a social network is true. For instance, Alice may tell Bob that she will be working until late, whereas she will actually go with her colleagues to have some beers. In this example, Bob has the (false) belief that Alice is working. Traditionally, in epistemic logic, the knowledge of agents consists on true facts, while beliefs represent plausible information that may be false [17]. For $\mathcal{KBL}_{\mathcal{RT}}$ we combine both modalities in one logic. In the following section we describe how to combine these two.
- *Time-stamped Epistemic Modalities*. Time-stamps are also part of the epistemic modalities K and B. Using time-stamps we can refer to the knowledge and beliefs of the agents at different points in time. For example, $B_{Bob}^{20:00}\mathit{loc}^{19:00}(Alice, work)$ means that Bob beliefs at $20{:}\,00$ that Alice's location at $19{:}\,00$ is work.
- *Occurrence of Events*. Being able to determine when an event has occurred allows users to define policies that are activated whenever someone performs an undesired event. Examples of these policies are: "if Alice unfriends Bob, she is not allowed

to send Bob a friend request" or "if Alice declines an invitation to Bob's party, then she cannot see any of the pictures uploaded during the party." We introduce $occurred^t(e)$ to syntactically capture the moment when a specific event e occurred.

2.2 Semantics

Real-Time Social Network Models We introduce formal models to reason about the states and evolution of social networks. These models leverage the information in the social graph [18] —the core data model in most social networks [19–21]. We extend social graphs, which include *agents* (or *users*) and the relationships between them, by adding for each agent a knowledge base, and the set of privacy policies that the agent has activated. We build upon a previous version of this framework [6], increasing substantially the expressiveness of privacy policies (see Section 3).

Definition 2 (Social Network Models). *Given a set of formulae $\mathcal{F} \subseteq \mathcal{F}_{\mathcal{KBL}_{\mathcal{RT}}}$, a set of privacy policies Π, and a finite set of agents $Ag \subseteq \mathcal{AU}$ from a universe $\mathcal{AU}$, a* social network model *(SNM) is a tuple $\langle Ag, \mathcal{A}, KB, \pi \rangle$, where*

- *Ag is a nonempty finite set of nodes representing the agents in the social network;*
- *$\mathcal{A}$ is a first-order structure over the SNM, consisting of a set of domains, a set of relations, a set of functions and a set of constants interpreted over their corresponding domain.*
- *$KB : Ag \to 2^{\mathcal{F}}$ is a function retrieving a set of (time-stamped) basic facts of an agent, which are stored in the agent's knowledge base; we write KB_i for $KB(i)$;*
- *$\pi : Ag \to 2^{\Pi}$ which returns the privacy policies of an agent; we write π_i for $\pi(i)$.*

In Def. 2, the shape of the relational structure $\mathcal{A}$ depends on the social network. We represent the connections—edges of the social graph—and the permission actions between social network agents as families of binary relations, respectively $\{C_i\}_{i \in \mathcal{C}} \subseteq Ag \times Ag$ and $\{A_i\}_{i \in \Sigma} \subseteq Ag \times Ag$ over the domain of agents. We use $\mathcal{D}$ to denote the set of domains. The set of agents Ag is always included in the set of domains. We use $\mathcal{SN}_{\mathcal{RT}}$ to denote the universe of all possible social network models.

Evolution of Social Network Models The state of a social network changes by means of the execution of *events* from the set EVT. For instance, in Facebook, users can share posts, upload pictures, like comments, etc. We use traces to capture the evolution of the social network. A *trace* is a finite sequence $\sigma = \langle (SN_0, E_0, t_0), (SN_1, E_1, t_1), \ldots, (SN_k, E_k, t_k) \rangle$ such that, for all $0 \leq i \leq k$, $SN_i \in \mathcal{SN}_{\mathcal{RT}}$, $E_i \subseteq EVT$, and $t_i \in \mathbb{T}$. We use $\mathbb{T}_\sigma = \{t \mid (SN, E, t) \in \sigma\}$ for the set of time-stamps of σ. We impose some conditions to traces so that they accurately model the evolution of social networks. We say that a trace is *well-formed* if it satisfies the following conditions:

1. Time-stamps are strictly ordered from smallest to largest, that is, for any i, j with $0 \leq i < j \leq k$ it follows that $t_i < t_j$.
2. Successor states are the result of events. We use $\to$ for the transition relation defined as $\to \subseteq \mathcal{SN}_{\mathcal{RT}} \times 2^{EVT} \times \mathbb{T} \times \mathcal{SN}_{\mathcal{RT}}$ ($\to$ can be specified using small step operational semantics as we show in [15] for an untimed version of this framework).

We write $SN_1 \xrightarrow{E,t} SN_2$ if SN_2 is the result of the set of events $E \in EVT$ happening in SN_1 at time t. We allow E to be empty, in which case $SN_1 = SN_2$.

3. For each $\xrightarrow{E,t}$ the set of events E must only contain independent events. Two events are independent if, when executed sequentially, their execution order does not change their resulting state. Formally, e and e' are independent whenever for every state SN_0 and time t, the state SN_2 and SN'_2 obtained as $SN_0 \xrightarrow{\{e\},t} SN_1 \xrightarrow{\{e'\},t} SN_2$ and $SN_0 \xrightarrow{\{e'\},t} SN'_1 \xrightarrow{\{e\},t} SN'_2$ satisfy that $SN_2 = SN'_2$. This definition can be easily extended to sets of events in the expected way.

We use $\mathcal{WFT}$ to refer to the set of well-formed traces. We assume that there is a function predecessor $pred : \mathbb{T} \to \mathbb{T}$ that takes a time-stamp and returns the previous time-stamp in the trace. Analogously, $next : \mathbb{T} \to \mathbb{T}$ returns the next time-stamp in the trace. Since the set of time-stamps is non-Zeno it is always possible to compute these functions.[3]

$\mathcal{KBL}_{\mathcal{RT}}$ formulae are very similar to $\mathcal{L}_n$ from epistemic logic [17]. There are two standard ways to define semantics of epistemic logics. First, one can define for every agent an undistinguishability relation between the *worlds* that the agent considers possible [17]. When considering traces of events, the framework typically used is *interpreted systems*. An alternative encoding, proposed by Fagin *et al.* [17][Section 7.3] consists in encoding the answer of epistemic queries from knowledge bases of accumulated facts. We follow this way of modelling knowledge here by equipping each agent with a knowledge base and defining the semantics of $\mathcal{KBL}_{\mathcal{RT}}$ formulae based on answers by these knowledge bases.

Extended Knowledge Bases An *Extended Knowledge Base* consists of a collection of $\mathcal{KBL}_{\mathcal{RT}}$ formulae, which represents the basic knowledge of the agent at a point in time. Epistemic derivations allow to answer whether a formula follows from the information stored in a knowledge base.

Derivations in EKBs. The EKB of an agent contains the explicit knowledge she acquired previously. Additional knowledge can be derived from the explicit pieces of information stored in these EKBs. Derivations can use formulae at a given point in time and at older times. We introduce the notion of *time window* (or simply, *window*) to determine how knowledge from the past can be used in a derivation. We write $\Gamma \vdash (\varphi, w)$ to denote that φ can be derived from Γ given a window w. We provide a set of deduction rules, DR, of the form given on the right meaning that, given the set of premises Γ, ψ can be derived with a window w from φ with a window w'.

$$\frac{\Gamma \vdash (\varphi, w')}{\Gamma \vdash (\psi, w)}$$

Definition 3. *A* timed derivation *of a formula φ with a window w, is a finite sequence of pairs $(\varphi_1, w_1), (\varphi_2, w_2), \ldots, (\varphi_n, w_n) = (\varphi, w)$ such that each φ_i, for $1 \leq i \leq n$, φ_i follows by an application of a deduction rule of DR whose premises φ_j, with $j < i$, have already been derived, and $w_j \leq w_i$.*

[3] We can assume that the predecessor of the initial time-stamp is the intial state itself, and similarly the next of the end of the trace returns the equal to itself.

Knowledge axioms		Belief axioms		Knowledge-Belief axioms	
A1	All tautologies of first-order logic	K	$(B_i^t\varphi \wedge B_i^t(\varphi \implies \psi)) \implies B_i^t\psi$	L1	$K_i^t\varphi \implies B_i^t\varphi$
A2	$(K_i^t\varphi \wedge K_i^t(\varphi \implies \psi)) \implies K_i^t\psi$	D	$\neg B_i^t\bot$	L2	$B_i^t\varphi \implies K_i^t B_i^t\varphi$
A3	$K_i^t\varphi \implies \varphi$	B4	$B_i^t\varphi \implies B_i^t B_i^t\varphi$		
A4	$K_i^t\varphi \implies K_i^t K_i^t\varphi$	B5	$\neg B_i^t\varphi \implies B_i^t\neg B_i^t\varphi$		
A5	$\neg K_i^t\varphi \implies K_i^t\neg K_i^t\varphi$				

Table 1: EKB axioms for a trace σ for each $t \in \mathbb{T}_\sigma$.

We now present the concrete derivation rules that allow to derive knowledge from the facts stored in EKBs. These rules extend axiomatizations of knowledge and belief with rules to deal with knowledge propagation through time.

Knowledge and Belief in EKBs. In our EKBs knowledge and belief can coexist. The common axiomatization of knowledge is **S5**. In [17], Fagin *et al.* provided an axiomatization for belief known as **KD45**, which includes the same set of axioms as **S5** —replacing K_i by B_i—except for the axiom $K_i\varphi \implies \varphi$ (A3). The reason is that beliefs do not need to be true—as required by A3 for knowledge. The requirement for beliefs is that an agent must have *consistent* beliefs, which is captured by $\neg B_i\bot$ (axiom D). To derive new knowledge, axioms from **S5** can be applied to formulas of the form $K_i\varphi$ and axioms from **KD45** for formulas of the form $B_i\varphi$. Additionally, derivations can also relate knowledge and beliefs, for which we use two axioms proposed by Halpern *et al.* in [12]: (L1) $K_i\varphi \implies B_i\varphi$ and (L2) $B_i\varphi \implies K_i B_i\varphi$. L1 states that when agents know a fact they also believe it, which is sound with respect to the definition of both modalities, since knowledge is required to be true (A3). Axiom L1 provides a way to convert knowledge to belief. L2 encodes that when agents believe a fact φ they know that they believe φ.

The previous axiomatizations are restricted to reasoning about a concrete time (or to timeless information), but we are interested in reasoning about the dynamic acquisition of knowledge in the changing world of online social networks. Consequently, we decorate all modalities with a time-stamp t, to explicitly capture the time at which an agent knows something and the time of occurrence of events and relations. Table 1 shows the complete list of axioms for a given $t \in \mathbb{T}_\sigma$.

To use these axioms in timed derivations we express them as deduction rules in Table 2. Note that all derivations that use these axioms use the same t and w. We add an explicit K_i^t to every formula in a user's EKB so that we can syntactically determine when some knowledge enters an EKB. Formally, we say that users in a trace σ are *self-aware* whenever for all $t \in \mathbb{T}_\sigma$ if $\varphi \in EKB_i^{\sigma[t]}$ then $\varphi = K_i^t\varphi'$. In what follows, we always assume that agents are self-aware.

Example 1. Consider the following EKB from a trace σ of an agent i at time t.

$$EKB_i^{\sigma[t]} \left\{ \begin{array}{c} K_i^t(\forall t' \cdot \forall j\colon Ag^{t'} \cdot event^{t'}(j, pub) \implies loc^{t'}(j, pub)) \\ K_i^t event^t(Alice, pub) \end{array} \right.$$

Knowledge deduction rules axioms		
$\dfrac{\varphi \text{ is a first-order tautology}}{\Gamma \vdash (\varphi, w)}$ (A1)	$\dfrac{\Gamma \vdash (K_i^t\varphi, w) \quad \Gamma \vdash (K_i^t(\varphi \implies \psi), w)}{\Gamma \vdash (K_i^t\psi, w)}$ (A2)	
$\dfrac{\Gamma \vdash (K_i^t\varphi, w)}{\Gamma \vdash (\varphi, w)}$ (A3)	$\dfrac{\Gamma \vdash (K_i^t\varphi, w)}{\Gamma \vdash (K_i^tK_i^t\varphi, w)}$ (A4)	$\dfrac{\Gamma \vdash (\neg K_i^t\varphi, w)}{\Gamma \vdash (K_i^t\neg K_i^t\varphi, w)}$ (A5)
Belief deduction rules		
$\dfrac{\Gamma \vdash (B_i^t\varphi, w) \quad \Gamma \vdash (B_i^t(\varphi \implies \psi), w)}{\Gamma \vdash (B_i^t\psi, w)}$ (K)	$\dfrac{}{\Gamma \vdash (\neg B_i^t\bot, w)}$ (D)	$\dfrac{\Gamma \vdash (B_i^t\varphi, w)}{\Gamma \vdash (B_i^tB_i^t\varphi, w)}$ (B4) $\quad \dfrac{\Gamma \vdash (\neg B_i^t\varphi, w)}{\Gamma \vdash (B_i^t\neg B_i^t\varphi, w)}$ (B5)
Premise deduction rule	**Knowledge-Belief deduction rules**	
$\dfrac{\varphi \in \Gamma}{\Gamma \vdash (\varphi, w)}$ (PREMISE)	$\dfrac{\Gamma \vdash (K_i^t\varphi, w)}{\Gamma \vdash (B_i^t\varphi, w)}$ (L1)	$\dfrac{\Gamma \vdash (B_i^t\varphi, w)}{\Gamma \vdash (K_i^tB_i^t\varphi, w)}$ (L2)

Table 2: EKB deduction rules for a trace σ for each $t \in \mathbb{T}_\sigma$.

In this EKB i can derive, using the axioms in Table 1, that Alice's location at time t is a pub, i.e., $loc^t(Alice, pub)$. Here we show the steps to derive this piece of information. We recall that quantifiers are unfolded when added to the knowledge base. For example, given formula $\varphi(x) : K_i^t\forall j\colon Ag^x \cdot event^x(j, pub) \implies loc^x(j, pub)$ and $\mathbb{T}_\sigma = \{t_0, t_1, \ldots, t\}$, the EKB contains $\varphi(t_0)\wedge\varphi(t_1)\wedge\ldots\wedge\varphi(t)$. The predicate $event^t(j, pub)$ means that j attended an event at time t in a pub. The predicate $loc^t(j, pub)$ means that j's location is a pub. Thus, the implication above encodes that: if i knows at t that an agent is attending an event in a pub at time t', then her location will be a pub. In this example, i knows at time t that Alice is attending an event at the pub, $event^t(Alice, pub)$. Since knowledge is required to be true, $event^t(Alice, pub)$ must be a true predicate. Hence, $K_i^t event^t(Alice, pub) \implies loc^t(Alice, pub)$ must also be present in $EKB_i^{\sigma[t]}$. Applying A2 to $K_i^t event^t(Alice, pub)$ and the previous implication we can derive $K_i^t loc^t(Alice, pub)$. □

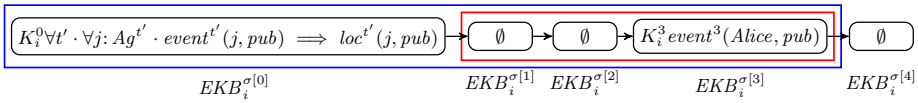
Handling time-stamps. Users can also use EKBs to reason about time. For instance, if Alice learns Bob's birthday she will remember this piece of information, possibly forever. Some other times information is transient and changes over time. Consider Alice, who shares with Bob a post including her location. Right after posting, Bob will know Alice's location—assuming she said the truth. However, after a few hours, Bob will not be certain about whether Alice remains in the same location. We denote the period of time in which some piece of information remains true as its *duration*.

Different pieces of information might have different durations. Duration also depends on the OSN, which can be designed in such a way that the effect of events disappears after some time. For example, in Snapchat messages last 10 seconds; in Whatsapp status messages last 24 hours; and in Facebook posts remain forever unless a user

removes them. We introduce the parameter w (see Section 1) to model the duration of the information. Using w we define the following deduction rule, which encodes a notion of duration-aware propagation of knowledge. Given $t, t' \in \mathbb{T}_\sigma$ where $t < t'$, the axiom (KR1) is shown on the right. The intuition behind KR1 is that some time in w is consumed every time knowledge is propagated. Consider that Alice knows at time 1 the formula φ, that is, $K^1_{Alice}\varphi$. Using KR1 in a derivation allows to derive that she knows φ at a later time, e.g., $K^5_{Alice}\varphi$. Note that this derivation requires w to be at least 4. The following example explains why.

$$\frac{\Gamma \vdash (K_i^t \varphi, w - (t' - t))}{\Gamma \vdash (K_i^{t'} \varphi, w)} \text{ (KR1)}$$

Example 2. Consider the following sequence of EKBs of an agent i from a trace σ where $\mathbb{T}_\sigma = \{0, \ldots, 4\}$.



Note that deriving Alice's location requires to combine knowledge from different knowledge bases at different times. This derivations use the knowledge recall rule KR1 with a large enough window. In the figure, the inner (red) rectangle marks the accessible knowledge for $w = 2$ and the outer (blue) rectangle for $w = 3$. In order for i to derive $loc^3(Alice, pub)$ she needs to combine knowledge from $EKB_i^{\sigma[0]}$ and $EKB_i^{\sigma[3]}$. Let $EKB_i^\sigma = \bigcup_{t \in \mathbb{T}_\sigma} EKB_i^{\sigma[t]}$. We first show how to construct a proof forwards, starting from the premises and a window of 0, and move forward increasing w until the inference can be performed. In particular, we show that $EKB_i^\sigma \vdash (K_i^3 loc^3(Alice, pub), w)$ for $w \in \mathbb{N}$. Applying the rule PREMISE with $w = 0$, we derive $EKB_i^\sigma \vdash (K_i^0 event^3(Alice, pub) \implies loc^3(Alice, pub), 0)$. Now we use KR1 to combine this knowledge with knowledge at time 3:

$$\text{(KR1)}\ \frac{EKB_i^\sigma \vdash (K_i^0 event^3(Alice, pub) \implies loc^3(Alice, pub), 0)}{EKB_i^\sigma \vdash (K_i^3 event^3(Alice, pub) \implies loc^3(Alice, pub), 3)}$$

This inference requires the window to be increased to 3. We apply PREMISE again to obtain $(EKB_i^\sigma \vdash K_i^3 event^3(Alice, pub), 3)$, which allows A2 to derive $(EKB_i^\sigma \vdash K_i^3 loc^3(Alice, pub), 3)$. This proof shows that i knows Alice's location provided that agents remember information for at least 3 units of time.

A window smaller than 3 makes this derivation impossible. We now construct the proof backwards, considering $w = 2$ to show that the derivation is impossible. We try to show that $EKB_i^\sigma \vdash (K_i^3 loc^3(Alice, pub), 2)$, which requires:

$$\text{(A2)}\ \frac{\begin{array}{c}(EKB_i^\sigma \vdash K_i^3 event^3(Alice, pub), 2) \\ EKB_i^\sigma \vdash (K_i^3 event^3(Alice, pub) \implies loc^3(Alice, pub), 2)\end{array}}{EKB_i^\sigma \vdash (K_i^3 loc^3(Alice, pub), 2)}$$

The first premise, $(EKB_i^\sigma \vdash K_i^3 event^3(Alice, pub), 2)$, trivially follows by PREMISE. To prove the second premise we first try move one step back using KR1: $EKB_i^\sigma \vdash (K_i^2 event^3(Alice, pub) \implies loc^3(Alice, pub), 1)$, but since there is no knowledge at

time 2, the previous statement cannot be proven. We apply again KR1 obtaining $EKB_i^\sigma \vdash (K_i^1 event^3(Alice, pub) \implies loc^3(Alice, pub), 0)$, which cannot be proven. Since the remaining window is 0, we have already accessed all knowledge that i remembers, and older EKBs cannot be accessed. This closes the proof. □

Belief propagation. Beliefs cannot be propagated as easily as knowledge because new beliefs may contradict current knowledge or beliefs of an agent. Instead of using timed derivations, we model agents that try to propagate beliefs if these beliefs are consistent, and discard them otherwise. We describe two kinds of agents: *conservative* and *susceptible*, but other criteria for choosing between incompatible beliefs are possible. We use the parameter β in the framework to denote the kind of agent. Conservative agents reject any new belief that contradicts their current set of beliefs, while susceptible agents always accept new beliefs that replace old believes if necessary to guarantee a consistent set of beliefs. Here we present a belief propagation algorithm which describes how agents behave when faced with a new belief.

Consider a trace σ with $\mathbb{T}_\sigma = \{t_0, \ldots, t_{n-1}, t_n\}$. We use the following notation $EKB_i^{\sigma[t_j,t_k]} = \bigcup_{t\in\{t_j,\ldots,t_k\}} EKB_i^{\sigma[t]}$. Also, we introduce the event $\mathsf{enter}(B_i^t\varphi)$ meaning that *belief φ enters i's knowledge base at time t*. The moment at which this event occurs identifies the moment when a belief is inserted in an agent's knowledge base, which is crucial to propagate beliefs. Given a belief $B_i^{t_n}\varphi$ that is about to enter $EKB_i^{\sigma[t_n]}$, i.e., $SN_{t_{n-1}} \xrightarrow{\mathsf{enter}(B_i^{t_n}\varphi),t_n} SN_{t_n}$, Algorithm 1 propagates the accumulated set of beliefs as long as they are inside the window w, and resolves conflicts according to β.

Lines 2-3 of Algorithm 1 construct a set Ψ of candidate beliefs to be propagated—according to w—together with the new belief that tries to enter i's EKB. The if block (lines 4-6) sorts Ψ according to β. In the foreach block (lines 7-11), we iterate over the sorted list of beliefs and add them to $EKB_i^{\sigma[t_n]}$ if they are consistent with the rest of knowledge and beliefs. It is easy to see that traversing beliefs from newest to oldest gives preference to newer beliefs in entering $EKB_i^{\sigma[t_n]}$, which corresponds to susceptible agents. In particular, $B_i^{t_n}\varphi$—the newest belief—will always enter the $EKB_i^{\sigma[t_n]}$ unless this belief contradicts actual knowledge. On the contrary, when sorting from oldest to newest, the older beliefs will have preference to enter $EKB_i^{\sigma[t_n]}$, thus, preventing new inconsistent beliefs to enter $EKB_i^{\sigma[t_n]}$, as required for conservative agents. In particular, $B_i^{t_n}\varphi$ will not be added to $EKB_i^{\sigma[t_n]}$ unless it is consistent with all the previous beliefs and knowledge. Finally, we always include the predicate $occurred^{t_n}(\mathsf{enter}(B_i^{t_n}\varphi))$ (line 12) so that the agent remembers that she was told $B_i^{t_n}\varphi$ —independently on whether she started to believe it. Note that consistency of $EKB_i^{\sigma[t_n]}$ in both cases is directly guaranteed by the inclusion condition in line 8.

Example 3. At 20:00 Alice sends a message to Bob indicating that she is at work, so $EKB_{Bob}^{\sigma[20:00]}$ contains $occurred^{20:00}(\mathsf{enter}(B_{Bob}^{20:00} loc^{20:00}(Alice, work)))$ and also $K_{Bob}^{20:00} B_{Bob}^{20:00} loc^{20:00}(Alice, work)$. At 22:00 Bob checks his Facebook timeline, and he sees a post of Charlie—who is a coworker of Alice—at 20:00 saying that he is with all his coworkers in a pub having a beer. Assuming that at 22:00 Bob still remembers his belief from 20:00 this new information creates a conflict with Bob's

Algorithm 1 Belief propagation

procedure BELIEF-PROPAGATION($EKB_i^{\sigma[t_n]}$, $B_i^{t_n}\varphi$, w, β)
  $\Psi \leftarrow \{K_i^{t_n} B_i^{t_n}\psi | occurred^t(\mathsf{enter}(B_i^t\varphi)) \in EKB_i^{\sigma[t_n - w, t_n]}$ where $t \in [t_n - w, t_n]\}$
  $\Psi \leftarrow \Psi \cup \{K_i^{t_n} B_i^{t_n}\varphi\}$
  if $\beta = susceptible$ **then** $[b_0, b_1, \ldots, b_n] \leftarrow sortNewestOldest(\Psi)$
  else if $\beta = conservative$ **then** $[b_0, b_1, \ldots, b_n] \leftarrow sortOldestNewest(\Psi)$
  end if
  foreach b **in** $[b_0, b_1, \ldots, b_n]$ **do**
    if $EKB_i^{\sigma[t_0,t_n]} \cup \{b\} \not\vdash B_i^{t_n}\bot$ **then**
      $EKB_i^{\sigma[t_n]} \leftarrow EKB_i^{\sigma[t_n]} \cup \{b\}$
    end if
  end foreach
  $EKB_i^{\sigma[t_n]} \leftarrow EKB_i^{\sigma[t_n]} \cup \{occurred^{t_n}(\mathsf{enter}(B_i^{t_n}\varphi))\}$
  return $EKB_i^{\sigma[t_n]}$
end procedure

beliefs. Note that information from Charlie's post is also taken as a belief since there is no way for Bob to validate it. If Bob is a conservative agent, then $EKB_{Bob}^{\sigma[22:00]} = \{K_{Bob}^{22:00} B_{Bob}^{22:00} loc^{20:00}(Alice, work)\} \cup \{occurred^{22:00}(\mathsf{enter}(B_{Bob}^{22:00} loc^{20:00}(Alice, pub)))\}$, meaning that the new belief is rejected. If Bob is a susceptible agent: $EKB_{Bob}^{\sigma[22:00]} = \{K_{Bob}^{22:00} B_{Bob}^{22:00} loc^{20:00}(Alice, pub)\} \cup \{occurred^{22:00}(\mathsf{enter}(B_{Bob}^{22:00} loc^{20:00}(Alice, pub)))\}$. Bob believes that Alice's location at time t ($20{:}00 \leq t < 22{:}00$) is work—due to belief propagation. After $22{:}00$, this belief does not propagate to avoid contradictions with the new belief $B_{Bob}^{22:00} loc^{20:00}(Alice, pub)$. □

Semantics of $\mathcal{KBL}_{\mathcal{RT}}$ The semantics of $\mathcal{KBL}_{\mathcal{RT}}$ formulae is given by the satisfaction relation $\models$. Given a well-formed trace $\sigma \in \mathcal{WFT}$, a window $w \in \mathbb{N}$, a time-stamp $t \in \mathbb{T}_\sigma$, agents $i, j \in Ag$, a finite set of agents $G \subseteq Ag$, formulae $\varphi, \psi \in \mathcal{F}_{\mathcal{KBL}_{\mathcal{RT}}}$, predicate symbols $c^t(i,j), a^t(i,j), p^t(\vec{s})$ where $c \in \mathcal{C}$ and $a \in \Sigma$, a domain $D \in \mathcal{D}$, an event $e \in EVT$, and a variable x, the *satisfaction relation* $\models \subseteq \mathcal{WFT} \times \mathcal{F}_{\mathcal{KBL}_{\mathcal{RT}}}$ is defined as follows:

$$
\begin{array}{lll}
\sigma \models occurred^t(e) & \text{iff} & (SN, E, t) \in \sigma \text{ such that } e \in E \\
\sigma \models \neg\varphi & \text{iff} & \sigma \not\models \varphi \\
\sigma \models \varphi \wedge \psi & \text{iff} & \sigma \models \varphi \text{ and } \sigma \models \psi \\
\sigma \models \forall t \cdot \varphi & \text{iff} & \text{for all } v \in \mathbb{T}_\sigma, \sigma \models \varphi[v/t] \\
\sigma \models \forall x : D^t \cdot \varphi & \text{iff} & \text{for all } v \in D^{\sigma[t]}, \sigma \models \varphi[v/x] \\
\sigma \models c^t(i,j) & \text{iff} & (i,j) \in C_c^{\sigma[t]} \\
\sigma \models a^t(i,j) & \text{iff} & (i,j) \in A_a^{\sigma[t]} \\
\sigma \models p^t(\vec{s}) & \text{iff} & p^t(\vec{s}) \in KB_e^{\sigma[t]} \\
\sigma \models K_i^t\varphi & \text{iff} & \bigcup_{\{t' | t' < t, t' \in \mathbb{T}_\sigma\}} KB_i^{\sigma[t']} \vdash (\varphi, w) \\
\sigma \models B_i^t\varphi & \text{iff} & \bigcup_{\{t' | t' < t, t' \in \mathbb{T}_\sigma\}} KB_i^{\sigma[t']} \vdash (B_i^t\varphi, w)
\end{array}
$$

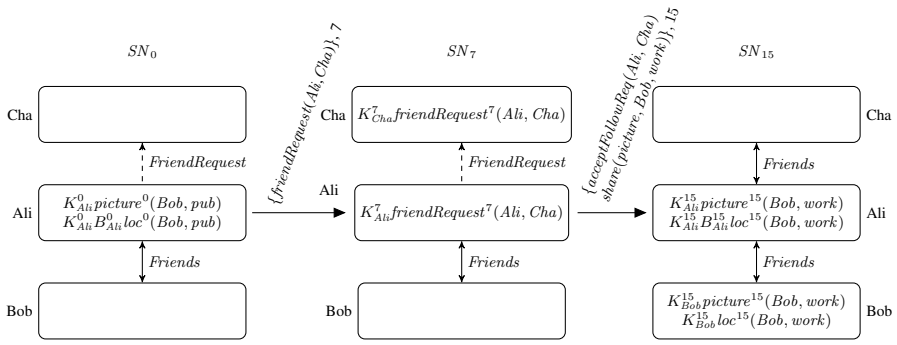


Fig. 1: Example of a Snapchat trace

Predicates of type $occurred^t(e)$ are true if the event e is part of the events that occurred at time t in the trace. $\forall t$ quantifies over all the time-stamps in the trace $\mathbb{T}_\sigma$, which is a finite set. For the remaining domains, $\forall x : D^t$, the substitution is carried out over the elements of the domain at a concrete time t. Remember that each individual domain D^t always contains a finite set of elements. However, the same domain at different points in time, e.g., D^t and $D^{t'}$, for any $t \neq t'$ might contain different number of elements. When checking connections $c^t(i, j)$ and actions $a^t(i, j)$ at time t, we check whether the corresponding relation— $C_c^{\sigma[t]}$ and $A_a^{\sigma[t]}$, respectively—of the SNM at time t contains the pair of users in question. Checking whether a predicate of type $p^t(\overrightarrow{s})$ holds is equivalent to looking into the knowledge base of the environment at time t. The environment's knowledge base contains all predicates that are true in the real world at a given moment in time. For example, "it is raining in Gothenburg at 19:00" $rain^{19:00}(Gothenburg)$ or "Alice's location at 20:00 is Madrid" $loc^{20:00}(Alice, Madrid)$. Determining whether an agent knows or believes a fact at a certain moment in time—i.e., $K_i^t\varphi$ or $B_i^t\varphi$—boils down to derivability from the union of all her EKBs for the given window w. This way of defining belief is based on the fact that agents are aware of their beliefs, recall axiom (L2) in Table 1.

Example 4 (Snapchat). In Snapchat users can perform two main events: (1) Connect through a friend relation; (2) share timed messages, which last up to 10 seconds with their friends. Fig. 1 shows an example trace for Snapchat with three agents $Ag = \{Alice, Bob, Charlie\}$. Since Ag does not change we avoid using the superindex indicating the time-stamp of the domain. The trace consists of three SNMs SN_0, SN_7 and SN_{15}, where the subindex indicates the time-stamp.

At time 0, Alice and Bob are friends, $friends^0(Alice, Bob)$, which is represented by including the pair $(Alice, Bob)$ in the relation $Friends^{\sigma[0]}$ in SN_0. Alice and Bob's friendship does not change along σ. Also, Alice is permitted to send a friend request to Charlie—depicted as an outgoing dashed arrow. Thus, $\sigma \models P^{Charlie}_{Alice} friendRequest^0$ holds. Finally, Alice knows that there is a picture of Bob at the pub, $picture^0(Bob, pub)$, and she *believes* that Bob is at the pub, $loc^0(Bob, pub)$. This is a belief because she cannot verify that the picture has not been modified or she cannot precisely identify

the location. However, the existence of $picture^0(Bob, pub)$ can be verified since it is a picture that Alice can see in the OSN. At time 7, Alice sends a friend request to Charlie. After the execution of the event both agents know $friendRequest^7(Alice, Charlie)$. Note that this event produces knowledge, because the agents can verify that the friend request has occurred. Finally, at time 15, Charlie accepts Alice's request and Bob shares a picture at work. Note that these two events are independent. After Bob accepts Alice's request $(Alice, Charlie) \notin FriendRequest^{\sigma[15]}$, and $(Alice, Charlie) \in Friends^{\sigma[15]}$. That is, Alice cannot send more friend requests to Charlie, and now they have become friends. Furthermore, both, Alice and Bob know that Bob shared a picture at work. In this case, Bob also knows that his location is work, but Alice only believes it.[4] The reason is that, unlikely Bob, Alice cannot confirm that Bob's location is work.

As on Snapchat messages last for up to 10 seconds, we can assume w.l.o.g. that all messages last 10 seconds, i.e., $w = 10$. Consequently, in σ, Alice remembers Bob picture from 0 to 10: $\sigma \models \forall t \cdot 0 \leq t \leq 10 \implies K^t_{Alice} picture^0(Bob, pub)$. Similarly, her belief about Bob location, $picture^0(Bob, pub)$, vanishes at time 10. Note also that, when Charlie accepts Alice's friend request, he still knows (or remembers) that Alice sent it. In Snapchat friend requests are permanent, but in our framework we can choose whether friend requests disappear after a few seconds. This can be done by requiring that the agent knows that a friend request occurred in order to accept it. In such a case, in σ, after time 18 Charlie would not be able to accept Alice's request. □

2.3 Model Checking $\mathcal{KBL}_{\mathcal{RT}}$

In this section, we show that the model checking problem for $\mathcal{KBL}_{\mathcal{RT}}$ is decidable.

Theorem 1. *The model checking problem for $\mathcal{KBL}_{\mathcal{RT}}$ is decidable.*

Proof. Let σ be a trace, φ a formula and w a window. Since all domains are finite, we unfold universal quantifiers $\forall x{:}\, D \cdot \varphi'$ and $\forall t \cdot \varphi'$ into a conjunction of formulas $\varphi'[v/x]$ for each element v in the domain D or in $\mathbb{T}_\sigma$. 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 every agent and at every step of the trace with either φ_k or $\neg\varphi_k$. The labelling proceeds from the earliest time-stamp on. We show only the epistemic operators here (see [22] for the complete proof):

- 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.
- First, we construct a set Δ where we instantiate all the axioms in Table 1 for each $t \in \mathbb{T}_\sigma$. The resulting set has size $|\Delta| = |\mathbb{T}_\sigma| \times 11$ (number of axioms in Table 1). Secondly, we instantiate KR1 (cf. Table 1), for w and for all $t, t' \in \mathbb{T}_\sigma$ such that $t > t'$ and $t - t' < w$. The resulting set of axioms has size $O(\sum_{n=1}^{|\mathbb{T}_\sigma|-1} n \times w)$. That is, all legal combinations of timestamps (n) times the window size (w). These

[4] For readability we omit $occurred^{15}(\mathsf{enter}(B^{15}_{Alice} loc^{15}(Bob, work)))$ in Fig. 1 which is included in $EKB^{\sigma[15]}_{Alice}$.

axioms are also included in Δ, which, consequently, contains a finite set of axioms. Finally, checking $K_i^t\psi_j$ and $B_i^t\psi_j$ requires one query to the epistemic engine for $\Delta, \bigcup_{\{t'|t'<t\in\mathbb{T}_\sigma\}} EKB_i^{\sigma[t]} \vdash \psi_j$. The previous query is equivalent to model checking a Kripke structure where relations are labelled with triples (i,t,w). Solving this problem is known to be decidable in PSPACE [17].

It is easy to see that the semantics of $\mathcal{KBL}_{\mathcal{RT}}$ is captured by this algorithm. □

2.4 Properties of the framework

Here we present a set of novel derived operators not present in traditional epistemic logics and we prove some properties of the framework (see [22] for the proofs).

To learn or not to learn — To believe or not to believe. In [6] we introduced a primitive modality $L_i\varphi$, to capture that i learns φ at the first moment at which $K_i\varphi$ becomes true. Here $L_i^t\varphi$ becomes a derived operator defined formally as: $L_i^t\varphi \triangleq \neg K_i^{pred(t)}\varphi \wedge K_i^t\varphi$. We can also model when users start to believe something, or *accept* a belief, as follows, $A_i^t\varphi \triangleq \neg B_i^{pred(t)}\varphi \wedge B_i^t\varphi$. Analogously we can express when users *forget* some knowledge or when they *reject* a belief. Intuitively, an agent forgets φ at time t if she knew it in the previous timestamp and in t she does not know φ, and, analogously, for reject. Formally, $F_i^t\varphi \triangleq K_i^{pred(t)}\varphi \wedge \neg K_i^t\varphi$, and $R_i^t\varphi \triangleq B_i^{pred(t)}\varphi \wedge \neg B_i^t\varphi$.

Temporal modalities The traditional temporal modalities $\Box$ and $\Diamond$ can easily be defined using quantification over timestamps as follows: $\Box\varphi(t) \triangleq \forall t \cdot \varphi(t)$, and $\Diamond\varphi(t) \triangleq \exists t \cdot \varphi(t)$, where $\varphi(t)$ is a formula φ which depends on t.

How long do agents remember? Agents remember according to the length of the parameter w, which can be seen as the size of their memory. Increasing agents memory could only increase their knowledge as stated in the following lemma.

Lemma 1 (Increasing window and Knowledge). *Given σ, $t \in \mathbb{T}_\sigma$ and $w, w' \in \mathbb{N}$ where $w \leq w'$, we have that: If $EKB_i^{\sigma[t]} \vdash (K_i^t\varphi, w)$, then $EKB_i^{\sigma[t]} \vdash (K_i^t\varphi, w')$.*

We can characterise how long agents remember information depending on w and β.

Lemma 2 (w knowledge monotonicity). *Given σ and $t \in \mathbb{T}_\sigma$. If $K_i^t\varphi \in EKB_i^{\sigma[t]}$ then for all $t' \in \mathbb{T}_\sigma$ such that $t \leq t' \leq t + w$ it holds $\sigma \models K_i^{t'}\varphi$.*

Perfect recall is obtained by choosing $w = \infty$ so agents never forget. Dualy, $w = 0$ models agents who do not remember anything. The parameter β also influences how beliefs are preserved in time. When $\beta = conservative$, memories about beliefs behave in the same way as knowledge. Similarly monotonicity results can be proven for beliefs as the lemmas above. For example, if $\beta = conservative$ then beliefs are preserved until these beliefs are forgotten—due to w—or contradict knowledge. Similarly, an agent with $\beta = susceptible$ rejects a belief when exposed to new contradictory beliefs. Therefore, the duration of their beliefs can be limited by an event introducing new beliefs in the EKBs. Other versions of β are possible, for example based on the reputation of the agent that emits the information. It is also possible to consider different w for different pieces of information. However, these extensions are out of the scope of this paper.

3 Writing Privacy Policies

We introduce here the language $\mathcal{PPL}_{\mathcal{RT}}$ for writing privacy polices: a restricted version of $\mathcal{KBL}_{\mathcal{RT}}$ wrapped with $[\![\ ]\!]_i^s$ (i is the owner of the policy, and s its starting time).

Definition 4 (Syntax of $\mathcal{PPL}_{\mathcal{RT}}$). *Given agents $i, j \in Ag$, a nonempty set of agents $G \subseteq Ag$, timestamps $s, t \in \mathbb{T}$, a domain $D \in \mathcal{D}$, a variable x, predicate symbols $c^t(i,j), a^t(i,j), p^t(\vec{s})$ where $c \in \mathcal{C}$ and $a \in \Sigma$, and a formula $\varphi \in \mathcal{F}_{\mathcal{KBL}_{\mathcal{RT}}}$, the* syntax of the real-time privacy policy language $\mathcal{PPL}_{\mathcal{RT}}$ *is inductively defined as:*

$$\begin{array}{ll} \delta ::= \delta \wedge \delta \mid \forall x \cdot \delta \mid [\![\neg\alpha]\!]_i^s \mid [\![\varphi \implies \neg\alpha]\!]_i^s & \gamma' ::= K_i^t\gamma \mid B_i^t\gamma \\ \alpha ::= \alpha \wedge \alpha \mid \forall x : D \cdot \alpha \mid \exists x : D \cdot \alpha \mid \psi \mid \gamma' & \psi ::= c^t(i,j) \mid a^t(i,j) \mid \mathit{occurred}^t(e) \\ \gamma ::= \gamma \wedge \gamma \mid \neg\gamma \mid p^t(\vec{s}) \mid \gamma' \mid \psi \mid \forall x \cdot \gamma & \end{array}$$

We use $\mathcal{F}_{\mathcal{PPL}_{\mathcal{RT}}}$ to denote the set of all privacy policies according to δ, and $\mathcal{F}^{\mathcal{R}}_{\mathcal{PPL}_{\mathcal{RT}}}$ the set positive formulae according to α, which we refer to as *restrictions*. As we show below restrictions appear always preceded by $\neg$. To determine whether a policy is violated in an evolving social network, we formalise the notion of conformance.

Definition 5 (Conformance Relation). *Given a trace $\sigma \in \mathcal{WFT}$, time-stamp $s \in \mathbb{T}_\sigma$, formulae $\delta \in \mathcal{F}_{\mathcal{PPL}_{\mathcal{RT}}}$ and $\alpha \in \mathcal{F}^{\mathcal{R}}_{\mathcal{PPL}_{\mathcal{RT}}}$, agent $i \in Ag$, domain $D \in \mathcal{D}$, and variable x, the* conformance relation $\models_C$ *is defined as follows:*

$$\begin{array}{lll} \sigma \models_C \forall x \cdot \delta & \text{iff} & \textit{for all } v \in D,\ \sigma \models_C \delta[v/x] \\ \sigma \models_C [\![\neg\alpha]\!]_i^s & \text{iff} & \sigma \models \neg\alpha \\ \sigma \models_C [\![\varphi \implies \neg\alpha]\!]_i^s & \text{iff} & s\sigma \models (\varphi \implies \neg\alpha) \end{array}$$

The definition is quite simple, especially compared to that of conformance of $\mathcal{PPL}_{\mathcal{T}}$ [6]. If the policy is quantified, we substitute in the usual way. The main body of the policy in double brackets is dealt with by simply delegating to the satisfaction relation.

Example 5. Alice decides to hide all her weekend locations from her supervisor Bob. She has a number of options how to achieve this using $\mathcal{PPL}_{\mathcal{RT}}$. If she wants to restrict Bob learning her weekend location directly when she posts it, she can define a policy stating that "if x is a time instant during a weekend, then Bob is not allowed to learn at x Alice's location from time x": $\delta_1 = \forall t \cdot [\![\mathit{weekend}(t) \implies \neg K^t_{Bob}\, \mathit{loc}^t(\mathit{Alice})]\!]^{\text{2017-10-20}}_{\mathit{Alice}}$, where $\mathit{weekend}$ is true if t represents a time during a weekend. This, however, is a very specialized scenario that captures only a small number of situations. Bob is, for example, free to learn Alice's location at any point not during the weekend, or at any point during the weekend when Alice's location is no longer up-to-date. We can consider a more precise policy concerning the learning of one's location: $\delta_2 = \forall t \cdot [\![\mathit{weekend}(t) \implies \neg\exists t' \cdot (K^{t'}_{Bob}\, \mathit{loc}^t(\mathit{Alice}))]\!]^{\text{2017-10-20}}_{\mathit{Alice}}$. Here, Bob is not allowed to learn Alice's location from a weekend, no matter when this information is learnt. □

Since checking conformance of $\mathcal{PPL}_{\mathcal{RT}}$ privacy policies reduces to model checking the given trace the following corollary follows directly from Theorem 1.

Corollary 1. *Checking conformance of $\mathcal{PPL}_{\mathcal{RT}}$ policies is decidable.*

4 Related Work and Concluding Remarks

Related Work Combining epistemic and timed reasoning has been previously studied. For example, [10] presents a logic for reasoning about actions and time. The logic includes a belief modality, actions, as well as time-stamps for atoms, modalities and actions. In our work we do not focus on reasoning about action and time but on defining dynamic privacy policies for OSNs. Also, [10] cannot reason about knowledge. Moses *et al.* [17] extends interpreted systems to reason about past and future knowledge. In [23] they extend K with a time-stamp $K_{i,t}$ allowing for reasoning about knowledge at different times, also having a similar predicate to our $occurred^t(e)$. However, our logic (unlike [23]) includes beliefs and associates time-stamps with both modalities and predicates, whereas [17] only uses time-stamps for knowledge. Additionally, [23] aims at modeling delays in protocols, whereas we want to express dynamic privacy policies for OSNs. Recently, Xiong *et al.* [11] presented a logic to reason about belief propagation in Twitter. The logic includes an (untimed) belief modality and actions, which are used in a dynamic logic fashion. Their models are similar to our untimed SNMs [24, 15]. Even though we do not include actions, we use time-stamps and knowledge modalities. Also, one of the main contributions of our paper is solving inconsistent beliefs.

Concluding Remarks We have presented a novel privacy policy framework based on a logic with time-stamps in events and epistemic operators. This framework extends [15, 24], which did not offer any support for time, and [6] which only had limited support due to the implicit treatment of time. A query in our framework starts by instantiating a number of epistemic axioms that handle knowledge, belief and time. Our proof system gives an algorithm to deduce the knowledge of agents acquired at each instant, and a model checking algorithm which can be used to check violations of privacy policies. The explicit time-stamps allow to derive learning and forget operators for knowledge, and accept and reject operators for beliefs. In our new framework we can define eternal OSNs like Facebook and ephemeral OSNs like Snapchat.

Two important avenues for future research are the following. First, the algorithm presented in this paper is not efficient enough for practical purposes, but serves as a formal foundation to develop provably correct efficient privacy violation detectors and enforcers, which can exploit specific details of each social network about how actions affect the knowledge of the agents involved. For instance, once the effect of the actions is fixed one can develop distributed algorithms that guarantee the same outcome as the direct algorithm proposed here. For example, tweets can only affect the knowledge of subscribers so all other users are unaffected. Second, once an effective system to check policy violations is in place, there are different possibilities that the OSN can offer. One is to enforce the policy by forbidding the action that the last agent executed (the action that leads to the violation). Another can be the analysis of the trace to assign blame (and affect the reputation) to the agents involved in the chain of actions. For example, the creator of a gossip or fake news may be held more responsible than users forwarding the gossip. A finer analysis of controllability allow more powerful algorithms that detecting which agents could have prevented the information flow that lead to the violation. Yet another possibility is to remove past events from the history trace of the OSN creating a pruned trace with no violation.

References

1. Lenhart, A., Purcell, K., Smith, A., Zickuhr, K.: Social media & mobile internet use among teens and young adults. millennials. Pew Internet & American Life Project (2010)
2. Madejski, M., Johnson, M., Bellovin, S.: A study of privacy settings errors in an online social network. In: PERCOM Workshops'12, IEEE (2012) 340–345
3. Johnson, M., Egelman, S., Bellovin, S.M.: Facebook and privacy: It's complicated. In: SOUPS'12, ACM (2012) 9:1–9:15
4. Liu, Y., Gummadi, K.P., Krishnamurthy, B., Mislove, A.: Analyzing facebook privacy settings: User expectations vs. reality. In: IMC'11, ACM (2011) 61–70
5. Madejski, M., Johnson, M.L., Bellovin, S.M.: The failure of online social network privacy settings. Technical report, Columbia University (2011)
6. Pardo, R., Kellyérová, I., Sánchez, C., Schneider, G.: Specification of evolving privacy policies for online social networks. In: TIME'16, IEEE (2016) 70–79
7. The Guardian: As fake news takes over Facebook feeds, many are taking satire as fact. www.theguardian.com/media/2016/nov/17/facebook-fake-news-satire [Accessed: 20/10/17].
8. The Guardian: How to solve Facebook's fake news problem: experts pitch their ideas. www.theguardian.com/technology/2016/nov/29/facebook-fake-news-problem-experts-pitch-ideas-algorithms [Accessed: 20/10/17].
9. The Guardian: Obama is worried about fake news on social media–and we should be too. www.theguardian.com/media/2016/nov/20/barack-obama-facebook-fake-news-problem [Accessed: 20/10/17].
10. van Zee, M., Doder, D., Dastani, M., van der Torre, L.W.N.: AGM revision of beliefs about action and time. In: IJCAI'15, AAAI Press (2015) 3250–3256
11. Xiong, Z., Ågotnes, T., Seligman, J., Zhu, R.: Towards a logic of tweeting. In: LORI'17. Volume 10455., LNCS (2017) 49–64
12. Halpern, J.Y., Samet, D., Segev, E.: Defining knowledge in terms of belief: The modal logic perspective. The Review of Symbolic Logic **2** (2009) 469–487
13. Fong, P.W.: Relationship-based access control: Protection model and policy language. In: CODASPY'11, ACM (2011) 191–202
14. Bruns, G., Fong, P.W., Siahaan, I., Huth, M.: Relationship-based access control: its expression and enforcement through hybrid logic. In: CODASPY'12, ACM (2012) 117–124
15. Pardo, R., Balliu, M., Schneider, G.: Formalising privacy policies in social networks. Journal of Logical and Algebraic Methods in Programming **90** (2017) 125–157
16. Snodgrass, R., Ahn, I.: Temporal databases. Computer **19**(9) (1986) 35–42
17. Fagin, R., Halpern, J.Y., Moses, Y., Vardi, M.Y.: Reasoning about knowledge. Volume 4. MIT press Cambridge (2003)
18. Erciyes, K.: Complex Networks: An Algorithmic Perspective. 1st edn. CRC Press, Inc. (2014)
19. FlockDB: A distributed fault-tolerant graph database: github.com/twitter/flockdb [Accessed: 20/10/17].
20. Bronson, N., Amsden, Z., Cabrera, G., Chakka, P., Dimov, P., Ding, H., Ferris, J., Giardullo, A., Kulkarni, S., Li, H., Marchukov, M., Petrov, D., Puzar, L., Song, Y.J., Venkataramani, V.: Tao: Facebook's distributed data store for the social graph. In: ATC'13. (2013) 49–60
21. Neo4j decreases development time-to-market for LinkedIn's Chitu App: neo4j.com/case-studies/linkedin-china/ [Accessed: 20/10/17].
22. Pardo, R., Sánchez, C., Schneider, G.: Timed Epistemic Knowledge Bases for Social Networks (Extended Version). ArXiv e-prints (2017)
23. Ben-Zvi, I., Moses, Y.: Agent-time epistemics and coordination. In: Logic and Its Applications. Volume 7750 of LNCS. Springer (2013) 97–108
24. Pardo, R., Schneider, G.: A formal privacy policy framework for social networks. In: SEFM'14. Volume 8702 of LNCS., Springer (2014) 378–392