Detecting and Predicting Privacy Violations in Online Social Networks

Özgür Kafalı · Akın Günay · Pınar Yolum

The final publication is available at Springer via http://dx.doi.org/10.1007/s10619-013-7124-8

Abstract Online social networks have become an essential part of social and work life. They enable users to share, discuss, and create content together with various others. Obviously, not all content is meant to be seen by all. It is extremely important to ensure that content is only shown to those that are approved by the content's owner so that the owner's privacy is preserved. Generally, online social networks are promising to preserve privacy through privacy agreements, but still everyday new privacy leakages are taking place. Ideally, online social networks should be able to manage and maintain their agreements through well-founded methods. However, the dynamic nature of the online social networks is making it difficult to keep private information contained.

We have developed $\mathcal{PROTOSS}$, a run time tool for detecting and predicting $\mathcal{PR}$ivacy vi$\mathcal{O}$la$\mathcal{T}$ions in $\mathcal{O}$nline $\mathcal{S}$ocial network$\mathcal{S}$. $\mathcal{PROTOSS}$ captures relations among users, their privacy agreements with an online social network operator, as well as domain-based semantic information and rules. It uses model checking to detect if relations among the users will result in the violation of privacy agreements. It can further use the semantic information to infer possible violations that have not been specified by the user explicitly. In addition to detection, $\mathcal{PROTOSS}$ can predict possible future violations by feeding in a hypothetical future world state. Through a running example, we show that $\mathcal{PROTOSS}$ can detect and predict subtle leakages, similar to the ones reported in real life examples. We study the performance of our system on the scenario as well as on an existing Facebook dataset.

1 Introduction

Preserving privacy has long been an important mission of Web systems. The general process of preserving privacy is through privacy agreements. Web systems announce their policies

Özgür Kafalı
Department of Computer Science, Royal Holloway, University of London
Egham, TW20 0EX, UK
E-mail: ozgur.kafali@rhul.ac.uk

Akın Günay · Pınar Yolum
Department of Computer Engineering, Bogazici University
TR-34342, Bebek, İstanbul, Turkey
E-mail: {akin.gunay, pinar.yolum}@boun.edu.tr

through privacy agreements. Users are expected to use the system only if they are comfortable with the agreement. In settings, where the Web system is a single locus of computation, carrying out privacy dealings through an agreement is reasonable. An example to such a setting is that of an e-commerce system. The e-commerce system announces to the user (through an agreement), whether it will share the account details or transaction details with third parties and to what extent. Knowing this, the user can decide whether this is an appropriate e-commerce system for her needs.

In online social networks (OSN), though, the loci of computation is distributed. The system that provides the social network service (such as Facebook) allows users to see each others' content, make comments, and even share the content with others. Hence, even if the entire content reside in a central location, multiple entities interact to post and share content. In such systems, it is difficult to maintain the privacy of users [4, 9]. Even if the system itself does not share the user information with other systems, other users on the social network can propagate a private content to others, for whom the content was not initially meant for. Or, other applications that benefit from the online social network can use private information for marketing or presentability purposes.

One famous story is that of a man who bought a diamond ring to his girlfriend to propose and the news showed up on his girlfriend's Facebook newsfeed before giving the ring, obviously ruining the event, and causing Facebook a big lawsuit [1]. Another famous example is how an individual's location can be identified from geographic information attached to the pictures they upload. While the online social network is not explicitly revealing the location information itself, through inferences the location is being deciphered without the users' knowledge and consent [14]. In both cases, the leaked information is not a simple privacy agreement violation. That is, the OSN operator does not immediately violate a privacy agreement. However, the relations among users and the inferences that are possible among the content make it possible for third parties to become aware of information that is actually private.

Hence, in systems that provide online social networks, even when a system operator correctly follows the privacy agreement that it announces, the privacy of users can easily be breached through interactions with other users. How can a user be sure that the content it publishes does not reach individuals that she does not want? The above discussion clearly shows that relying on static privacy agreements and hoping for the best does not solve the problem. On the contrary, there is a great need for a dynamic approach that will (1) actively detect if a person's privacy is being breached at run time and (2) reason on the user's expectations to see if any unspecified situations might jeopardize her privacy.

Actively detection privacy breaches will require tools that monitor how an OSN is evolving to signal any situations that will cause privacy violations and preferably even signal future situations that might be possible and cause privacy violations later. Here, we understand a privacy violation as a situation where a private content is being shared with certain individuals that should not be shown the content. An obvious reason for a privacy violation is the non-compliance of the OSN operator with the privacy agreement. The OSN operator can detect and correct such cases by aligning its behavior with its privacy agreements. A more subtle reason for privacy violations is the conflict that can easily exist between various privacy agreements. In such cases, only one of the agreements can be satisfied and the second privacy agreement is violated, causing a privacy violation. Following the above example, a tool that could warn the future groom (and the OSN) that the OSN's working could lead to a privacy violation, before it even took place would have been tremendously useful. This would imply that the tool would check the current state of the system, possible future states

in which the system can evolve in, and decide if any of these states are potentially harmful for the user.

Reasoning on the user's expectations will require understanding the user's constraints and following up on possible constraints that have not been explicitly specified. Following the above example, it is not sufficient to manually configure the system to hide the location information from pictures. The system should understand the constraint enough to generalize that any media that gives away the location information need to be handled delicately.

Accordingly, we propose $\mathcal{PROTOSS}$, a run time tool to detect and predict possible privacy breaches in online social networks. The main technique underlying our approach is model checking, which given a model of the system verifies whether certain properties hold [8]. Our system model represents the privacy agreements of an OSN with its users and the relations between the users formally. $\mathcal{PROTOSS}$ uses semantic information such as a content ontology and inference rules to reason on users' expectations. Using $\mathcal{PROTOSS}$, we can check interesting properties such as whether a user's content may reach to some individuals who are not intended to see that content in the first place or whether private information of a user may be revealed to third parties due to various relations in the social network. We show that $\mathcal{PROTOSS}$ can detect and predict subtle information leakages that are not easy to detect in conventional OSNs and that it can even suggest certain leakages for contents that are not declared private in the first place. We demonstrate these over scenarios. We then experiment with a Facebook dataset to study the applicability of our approach.

The rest of this paper is organized as follows. Section 2 provides the necessary technical background on commitments, model checking, and ontologies. Section 3 describes our privacy architecture with its main components. Section 4 illustrates a social network with possible instantiations of components and provides intuitive scenarios that can take place. Section 5 provides the details of $\mathcal{PROTOSS}$, which is used to detect and predict privacy violations. Section 5.1 evaluates our approach on the given scenarios and presents performance results. Finally, Section 6 discusses our contributions and results with respect to the literature.

2 Technical Background

There are three important techniques underlying our approach. The first one is the abstraction of commitment among individuals when carrying out agreements [21]. The second one is checking models of systems to verify whether certain properties hold in the system [8]. The third one is the use of ontologies to define the semantics of concepts related to OSNs and privacy and reason about their relationships [13, 19].

2.1 Commitments

A *commitment* is an agreement from one party to another to bring about a certain property, if a specified condition is achieved [21]. For instance, a social network operator may commit to one of its users *Charlie* to share his location with another user *Patty*, only if *Charlie* and *Patty* are friends. More formally, a commitment is denoted as $C(debtor, creditor, antecedent, consequent)$, which states that the $debtor$ is committed to the $creditor$ to bring about the $consequent$, if the $antecedent$ is achieved [26]. We can represent the above case the formal notation by generalizing it to every friend of *Charlie* as follows: $C(operator,$

$charlie$, $friend(charlie, X)$, $shareLocation(charlie, X)$), where the social network operator ($operator$) is the debtor, $charlie$ is the creditor, the antecedent $friend(charlie, X)$ represents $charlie$ and some another user X are friends and the consequent $shareLocation(charlie, X)$ represents that the location information of $charlie$ is shared with X.

A commitment is not just a static representation of an agreement. Instead, it is an active entity that reflects the current state of the underlying agreement. In order to achieve this, a commitment is associated with a state that evolves over time in coordination with the status of the represented agreement [26]. A commitment is created in *conditional* state, in which neither the antecedent nor the consequent of the commitment is achieved. When the antecedent of the commitment is achieved, the state of the commitment changes to *active*. If the consequent of the commitment is satisfied, then the state of the commitment changes to *fulfilled*. On the other hand, if the consequent of the commitment fails while the commitment is active, then the state of the commitment changes to *violated*. Let us discuss the meaning of these states from the debtor and creditor's point of view using the above example. In the above example the commitment from the *Operator* to *Charlie* is initially in conditional state. If *Charlie* creates a friend relation with somebody, say *Patty*, then the commitment becomes active. From now on, it is the responsibility of the *Operator* to provide the location of *Charlie* to *Patty* and fulfill its commitment. On the other, if the *Operator* fails to provide the location of *Charlie* to *Patty*, then it violates its commitment.

Commitments provide a suitable mechanism to represent agreements between parties in a dynamic manner. Our example represents a basic agreement with a single term between two parties. However, commitments can be used to handle more complex situations, where agreements can have multiple terms. Additionally, by combining multiple commitments, interactions between more than two parties can also be captured [26].

2.2 Model Checking

Model checking is a computational method to automatically verify whether a certain property holds in a given system [8,15]. The system under consideration is modeled as a state transition graph in some formal language and the property that is aimed to be verified is represented as a logic formula in a suitable language, such as linear temporal logic (LTL) or computation tree logic (CTL) [11]. Given the system model and the logic formula of the investigated property, a model checking algorithm verifies whether the system model satisfies the property.

In this work we use NuSMV, a state of the art model checker that is based on the use of binary decision diagrams [7]. A NuSMV model defines a set of variables and how these variables evolve according to the possible executions of the modeled system. For instance, a variable may represent that two users are friends in a social network and evolution of this variable can be modeled according to the operations provided on this relation by the modeled social network. In other words, a NuSMV model defines the underlying operational mechanism of the considered system. Once there is such a model, NuSMV can be used to verify certain properties of the model. For instance, a property of the social network about privacy could be that the location of a user is not revealed to other users that are not friends.

NuSMV uses CTL to represent properties that are aimed to be verified. CTL is a branching time logic, where the future is modeled as a tree structure in which each branch corresponds to a possible different future. CTL formulas are built up from a set of propositional variables, the usual logic connectives and a set of temporal modal operators. The first type of temporal operators are A and E, which quantify over paths. A stands for *all* and means

that the quantified formula has to hold on all paths. E stands for *exists* and means that the quantified formula has to hold at least on one path. The other four temporal operators X, F, G and U are specific to a single path. X stands for *next* and means that its bounded formula has to hold at the next state of the given path. F stands for *eventually* and means that the bounded formula has to hold eventually at some future state(s) of the given path. G stands for *globally* and means that the bounded formula has to hold at all future states of the given path. Finally, U stands for *until* and it is the only binary operator which means that the first formula bounded to U has to hold until the second formula starts to hold.

2.3 Ontologies

An ontology is a conceptualization of a particular domain [13, 19]. An ontology enables specification of domain concepts and their relations semantically. The concepts are defined using their properties. The relations enable concepts to be tied together. A common relation between concepts is the *isA* relation, which denotes that one concept is a type of another concept (e.g., a cat *isA* animal). With only *isA* relation, one can define a hierarchy over the concepts. Obviously, more relations can be added to the ontology to make the domain representation rich. Once a domain is represented as an ontology, one can reason on it and make inferences. For example, a rule that applies to animals should also apply to cats, since cat is a type of an animal based on the ontology.

From a privacy perspective, content related to privacy can be thought of as a domain and represented as an ontology. For example, if a user only wants its media contents to be shown to other users, an ontological reasoning can discover that pictures are types of media and would show that information, whereas someone's e-mail address is not a type of media and therefore would not be shown to others.

3 Privacy-Aware OSN Architecture

We are interested in online social networks (OSNs) that are administered by an OSN operator. Each user can post content as it sees fit. The content could vary. One can post personal information such as her location, the people she is with, and so on as well as links to news, jokes and so forth. Our primary aim in this work is the first set of information since we are interested to see how private information can float in the system.

Since it is a social network, users are related to each other. As in newer social networks, users can be related to each other through different relations. For example, Charlie could be a friend of Sally but a colleague of Linus. These relations identify how much and of what type of content would be shared with other users. For example, Charlie would share his whereabouts with his friends, but may not want to share this with colleagues. Essentially, the OSN operator is responsible for ensuring that these expectations are met. That is, the OSN operator is supposed to ensure that only the users with the right privileges are shown private content.

Fig. 1 demonstrates the architecture that we use to check privacy agreements. The circles represent the users, who provide content to the system. The users are connected to each other through relations. Among each user and the OSN operator, there exists a privacy agreement. This is an agreement that contains clauses about which relations are entitled to which privileges. For example, an agreement between Charlie and the OSN operator can state that all friends of Charlie are entitled to see his location. This is not a static agreement. That is,

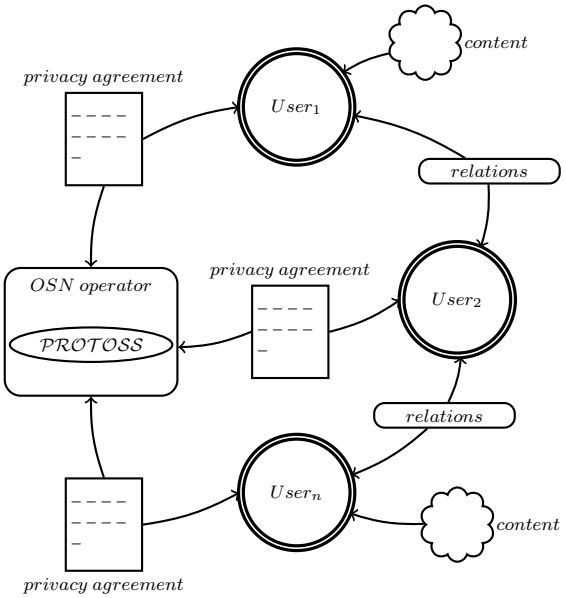

Fig. 1 Overview of a privacy-aware OSN architecture.

as Charlie creates more relations with other users or creates different types of content, this privacy agreement is updated accordingly.

OSN operator is responsible for realizing the clauses in the privacy agreement. However, it might be the case that based on the relations or newly added agreements, some privacy agreements cannot be honored, thereby leading to a privacy violation. For this reason, OSN operator needs a privacy checker to check as needed whether it can honor a certain clause in a privacy agreement. The privacy checker we propose here is $\mathcal{PROTOSS}$.

$\mathcal{PROTOSS}$ uses the network information as well as the privacy agreements to decide whether the agreements can be honored in the system. For example, assume that Charlie, i) wants his friends to see his location but, ii) does not want any of his colleagues to see his location and OSN agrees to honor this through a privacy agreement. Assume that during the execution of the system, Charlie identifies Linus both as a colleague and a friend. This puts OSN in a conflicting situation since if he lets Linus see Charlie's location it will honor part of the agreement but violate the second part. If the OSN does not show the location, then it will violate the first part but honor the second part. In this case, OSN should let Charlie know of the situation and maybe let him decide what to do.

$\mathcal{PROTOSS}$ can be used in two modes. First, it can be used to *detect* privacy violations by the OSN operator in the current state of the world based on the available information. The detection of violations is expected to lead to corrections in the system. Second, it can be used to *predict* privacy violations by considering possible futures of the world. That is, it might be the case that even though there is no privacy violation right now, if certain relations hold in the future (e.g., two users become friends), then a privacy violation might occur. It is useful to know under which circumstances this violation would take place. Prediction can be initiated either by the user or the OSN. The user might create a hypothetical future and query $\mathcal{PROTOSS}$ to see if in this possible future state, a violation would take place. Alternatively, OSN operator can use $\mathcal{PROTOSS}$ to generate a possible future state in which a violation

would take place. In this case, the generated future state would contain a set of relations that need to hold for a violation to take place. The prediction of violations would help users or the OSN to avoid certain world states before they even take place.

4 Running Example

In this section, we present an example OSN in Fig. 2 that we use as a running example in the rest of the paper. In the following, variables X, Y, Z are over users and L, I, V are over location, picture and video contents, respectively.

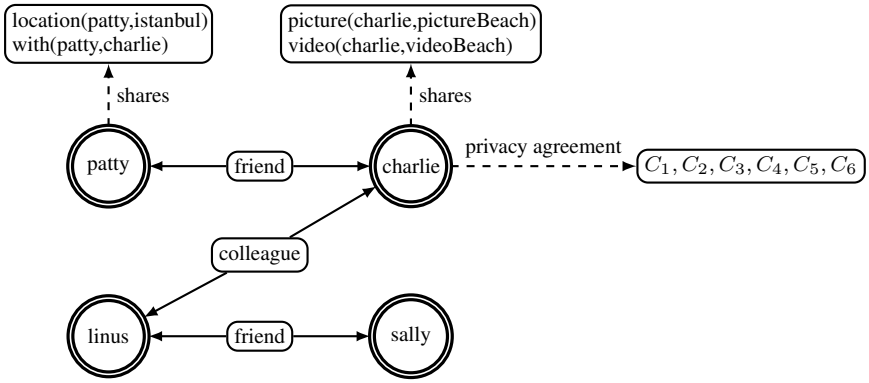

Fig. 2 Initial state of the example online social network including users, relations, shared content, and privacy agreements.

i. Users: In the example OSN we have four users; $charlie$, $patty$, $sally$ and $linus$. The users are represented by double-lined circles in Fig. 2.

ii. Relations: The users of the example OSN can initiate or terminate relations among themselves. This is a typical property of online social networks [6]. We assume that there are two types of relations:

- $friend(X, Y)$: Users X and Y are friends.
- $colleague(X, Y)$: Users X and Y are colleagues.

In Fig. 2, the relations are represented by the bi-directional edges between the users, which are labeled as *friend* and *colleague* depending on the type of relation between the users. For instance, while users $charlie$ and $patty$ are in a $friend$ relation, $charlie$ and $linus$ are in a $colleague$ relation. We assume that if a relation is not initiated, then it does not hold (e.g., $charlie$ and $sally$ are not friends in the example OSN).

iii. Content: To complement the relations in the example OSN, we have the following types of content that can be shared by the users:

- $location(X, L)$: User X is at location L.
- $with(X, Y)$: User X is with user Y.
- $picture(X, I)$: User X posts a picture I.
- $video(X, V)$: User X posts a video V.

In Fig. 2, the content that is shared by a user is represented by a rectangle, which includes the listings of the shared content instances. Each rectangle has an incoming dashed directed edge from a user, which is labeled as *shares*, to indicate the user who shares the content. For instance, $patty$ shares the content that she is in Istanbul (i.e., $location(patty, istanbul)$).

iv. OSN operator: The example OSN in Fig. 2 is managed by a single OSN operator (not represented in the Fig. 2 for brevity). In particular, we are interested in how the OSN operator manages user access to the shared content. In this context, we assume that the operational behavior of the OSN operator is defined by a set of conditional behavior rules as presented below. In the following, we define a specific $visible$ predicate for all types of contents (e.g., $visibleLocation$, $visiblePicture$, etc.), which states that the specified content is visible (i.e., accessible) to a defined user.

- B_1: $visibleWith(with(X, Y), Z) \leftarrow friend(X, Z) \vee friend(Y, Z)$: This rule states that if the user Z is a friend of either user X or Y then the OSN operator reveals the content that X and Y are together to Z.
- B_2: $visibleLocation(location(X, L), Y) \leftarrow friend(X, Y)$: This rule states that if the user Y is a friend of user X, then the OSN operator will show X's location L to Y.
- B_3: $visiblePicture(picture(X, I), Y) \leftarrow friend(X, Y)$: This rule states that if the user Y is a friend of user X, then the OSN operator will show X's picture I to Y.
- B_4: $visibleVideo(video(X, V), Y) \leftarrow friend(X, Y)$: This rule states that if the user Y is a friend of user X, then the OSN operator will show X's video V to Y.

For a particular example, consider B_3 in the case of users $charlie$ and $patty$. $charlie$ and $patty$ are friends in the OSN (i.e., $friend(charlie, patty)$). Besides, $charlie$ shares the picture $pictureBeach$ (i.e., $picture(charlie, pictureBeach)$). Hence, the OSN operator reveals the content $pictureBeach$ to $patty$.

We assume that a particular content is visible to a given user only if one of the rules $B_1 - B_4$ is applicable considering that user and the content. Otherwise, the OSN does not allow the user to access the content. For instance, in the case of $charlie$ and $linus$, the picture of $charlie$ is not revealed to $linus$ by the OSN operator, since none of the rules is applicable.

v. Privacy agreements: A privacy agreement between a user and the OSN operator is a set of commitments made from the OSN operator to the user about how the content shared by the user is revealed to the other users. For brevity, in our running example we present only the privacy agreement of $charlie$ that includes the following commitments:

- C_1: $C(osn, charlie, friend(charlie, Y), visiblePicture(picture(charlie, I), Y))$: The OSN operator commits to $charlie$ that his friends will be able to see his pictures.
- C_2: $C(osn, charlie, friend(charlie, Y), visibleWith(with(charlie, Z), Y))$: The OSN operator commits to $charlie$ that his friends will be able to see who he is with.
- C_3: $C(osn, charlie, friend(charlie, Y), visibleLocation(location(charlie, L), Y))$: The OSN operator commits to $charlie$ that his friends will be able to see where he is.
- C_4: $C(osn, charlie, colleague(charlie, Y), \neg visiblePicture(picture(charlie, I), Y))$: This commitment is slightly different from the first tree. Here, the OSN operator commits to $charlie$ that his colleagues will not see his pictures.
- C_5: $C(osn, charlie, colleague(charlie, Y), \neg visibleWith(with(charlie, Z), Y))$: Again, the OSN operator commits to $charlie$ that his colleagues will not see who he is with.
- C_6: $C(osn, charlie, colleague(charlie, Y), \neg visibleLocation(location(charlie, L), Y))$: The OSN operator commits to $charlie$ that his colleagues will not see where he is.

In Fig. 2, the privacy agreement of $charlie$ is represented by the rectangle that includes the listings of the commitment instances and has an incoming dashed directed edge from $charlie$, which is labeled as *privacy agreement*. For brevity, we write only the IDs of the commitments as listed above.

At this point, note the difference between the behavior rules we define earlier and a privacy agreement. The behavior rules state how the OSN operator is going to act (based on its design and implementation) whereas the privacy agreement states what the OSN operator promises to a user. The behavior rules are private to the OSN operator, whereas the privacy agreement is shared between a user and the OSN operator. Ideally, the behavior rules and the privacy agreement should be compatible, such that the OSN operator would act in a way that honors its agreement with the user. However, in reality this might not be the case, leading to privacy violations. In this context, next we describe several scenarios that might occur in the example OSN. We first present five scenarios focusing on the detection of privacy violations (Scenarios 1-5). In these scenarios, the OSN operator checks whether it is fulfilling its commitments (i.e., privacy agreements) to the user $charlie$ in the current state of the OSN. Then we present two scenarios focusing on the prediction of privacy violations before they actually occur (Scenarios 6-7). In these scenarios, the user $charlie$ tries to predict possible breaches of his privacy by making assumptions about future relations of the other users.

Scenario 1 According to the commitment C_4 in the privacy agreement of $charlie$, pictures of $charlie$ should not be revealed to his colleagues. Hence, $linus$ should not be able to see $charlie$'s picture $pictureBeach$. Given the network setting in Fig. 2, the aim of the OSN operator is to detect whether it is possible for $linus$ to see $pictureBeach$.

Scenario 2 According to the commitment C_6 in the privacy agreement of $charlie$, location of $charlie$ should not be revealed to his colleagues. Hence, $linus$ should not be able to see $charlie$'s location. In the setting of Fig. 2, $charlie$ does not share his location, but $patty$ does (indirectly), since she shares that she is with $charlie$ and she is in Istanbul. From this information it is easy to conclude that $charlie$ is in Istanbul too. Hence, in this setting, the aim of the OSN operator is to detect whether it is possible for $linus$ to find out $charlie$'s location.

Scenario 3 As $charlie$ stated in his privacy agreement, he does not want his colleagues to view his pictures (C_4). However he has not made any statement about his videos (knowingly or unknowingly). Is it possible to make further reasoning to infer that videos are by nature similar to pictures and if any videos of $charlie$ are being seen by colleagues, might it be worthwhile to notify him?

Scenario 4 Assume that $charlie$ meets $sally$ in Istanbul and adds her as a friend in the OSN. Hence, the OSN evolves into a state we represent in Fig. 3. The aim of the OSN operator is to detect whether $charlie$'s picture is visible to $linus$ in this new state of the OSN.

Scenario 5 In the new state of the OSN, after the new friend relation between $charlie$ and $sally$ was initiated, $sally$ shares that she is in Istanbul and she is with $charlie$. Is it possible for $linus$ to learn $charlie$'s location after $sally$ shares this information, which causes violation of $charlie$'s privacy agreement?

In the following two scenarios we go back to the initial state of the OSN (i.e., $charlie$ and $sally$ are not friends yet) and $charlie$ tries to predict possible future breaches of his

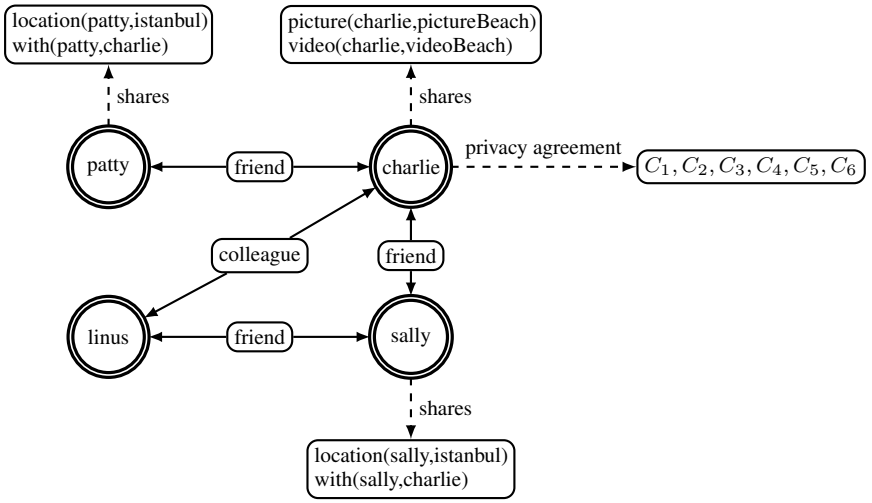

Fig. 3 Evolved state of the example online social network after $charlie$ adds $sally$ as a friend.

privacy depending on the evolution of relations between the users. In this context $charlie$ uses his own knowledge about the OSN (i.e., his existing relations), which we represent in Fig. 4. Besides his own knowledge, he can make also assumptions about the possible relations of the other users, if such information exists via any source. For instance, $charlie$ may assume $patty$ and $linus$ are not friends. $charlie$ may make this assumption since he is a friend of $patty$ and knows she is not friend with $linus$.

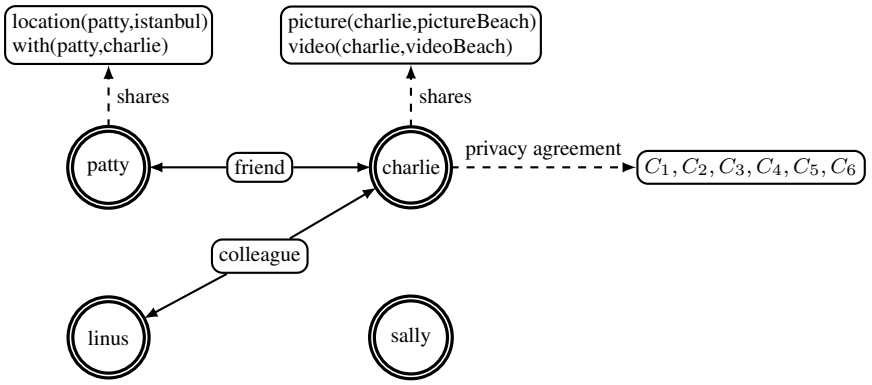

Fig. 4 $charlie$'s knowledge about the OSN in the initial state (i.e., before adding $sally$ as a friend).

Scenario 6 $charlie$ is a cautious user and desires to find out what would it take for $linus$ to see his pictures. That is, what relations in the OSN need to be initiated between the users of the OSN in the future for this information to leak? In this scenario, $charlie$ chooses not to make any assumptions about the relations of the other users.

Scenario 7 $charlie$ wants to add $sally$ as a friend. But he is concerned that this may cause $linus$ to see his pictures. Therefore, before adding $sally$ as a friend, he wants to find out

```
MODULE commitment(ant, cons)

CONSTANTS CONDITIONAL, ACTIVE,
          FULFILLED, VIOLATED;
DEFINE
 status:=
  case
   !ant: CONDITIONAL;
   ant & cons = Undetermined: ACTIVE;
   ant & cons = True: FULFILLED;
   ant & cons = False: VIOLATED;
  esac;

MODULE neg_commitment(ant, cons)
CONSTANTS CONDITIONAL, ACTIVE,
          FULFILLED, VIOLATED;
DEFINE
 status:=
  case
   !ant: CONDITIONAL;
   ant & cons = Undetermined: ACTIVE;
   ant & cons = False: FULFILLED;
   ant & cons = True: VIOLATED;
  esac;
```

Listing 1 Definition of the commitments module in NuSMV.

whether his pictures would be visible to $linus$ if he adds $sally$ as a friend. In this scenario, $charlie$ also assumes $patty$ and $linus$ are not friends.

5 $\mathcal{PROTOSS}$ for Detecting and Predicting Violations

The reasoning necessary for the scenarios above is done by $\mathcal{PROTOSS}$. $\mathcal{PROTOSS}$ has two important components: *commitment* component, which is responsible from checking the state of the commitments (i.e., privacy agreements) and the *semantic* component, which is responsible for performing semantic reasoning.

Commitment Component: The $\mathcal{PROTOSS}$ engine uses NuSMV model checker as a core component. However, NuSMV is by itself not capable of checking models with commitments in them. Hence, we have first introduced a commitment module into the NuSMV model checker, based on Telang and Singh's work [22].

Listing 1 provides the commitments module that we use to describe commitments. This module reflects the evolution of commitment states according to the values of the antecedent and the consequent as described in Section 2.1. We have modeled two types of commitments in terms of the fulfillment of the consequent. The consequent can have three values: (i) *Undetermined* means that the value of the consequent is not known at that point in time, (ii) *True* means that it is satisfied, and (iii) *False* means that it is unsatisfied. Accordingly, the first commitment type corresponds to a promise such that the commitment is violated if the consequent is unsatisfied. These commitments are labeled as $commitment$ in the commitment module of $\mathcal{PROTOSS}$.

```
c3: commitment(friend_charlie_patty,
```

```
    visible_location_charlie_patty);
```

The listing above shows a commitment instance of C_3 in our running example in Section 3, in which the OSN operator commits to $charlie$ to show his location to $patty$ (visible_location_charlie_patty), if $charlie$ and $patty$ are friends (friend_charlie_patty). Initially when $charlie$ and $patty$ are not friends the commitment is conditional (i.e., antecedent friend_charlie_patty does not hold) and the OSN operator should not show $charlie$'s location to $patty$. When the friend relation is initiated, the commitment is in active state. Note that the value of the consequent visible_location_charlie_patty is unknown yet. Once the OSN operator reveals the location of $charlie$ to $patty$, the value of the consequent visible_location_charlie_patty is true and the commitment is fulfilled. On the other hand, the commitment is violated if the OSN fails to show $charlie$'s location to $patty$ (i.e., consequent visible_location_charlie_patty is false).

On the other hand, the second commitment type corresponds to a prohibition where the commitment is violated when the consequent is satisfied. These commitments are labeled as $neg\_commitment$ in the commitment module of $\mathcal{PROTOSS}$.

```
c6: neg_commitment(
      colleague_charlie_linus,
      visible_location_charlie_linus);
```

The listing above shows such a commitment instance of C_6 in our running example in Section 3, in which the OSN operator commits to $charlie$ to not to show his location to $linus$ (visible_location_charlie_linus), if $charlie$ and $linus$ are colleagues (colleague_charlie_linus). This commitment is fulfilled when the location of $charlie$ is not visible to $linus$ (i.e., visible_location_charlie_linus is false) and violated if the location of $charlie$ is visible to $linus$ (i.e., visible_location_charlie_linus is true).

Semantic Component: An important component of $\mathcal{PROTOSS}$ is its semantic reasoning to derive new information from existing knowledge. In order to do this, it makes use of an ontology that captures the definitions and semantic relations of contents. The domain represented in the ontology corresponds to the domain that a user may share with others, such as her location, pictures, and so on. The ontology would capture facts such as a person's location is a type of personal information, or a person's picture is a type of her media content.

The ontology is used in two ways in $\mathcal{PROTOSS}$. The first one is to refine commitments. For example, if OSN commits to a user not to share any of her information, then a quick reasoning on the ontology would yield that neither a person's location nor her pictures can be shared. Second, undefined privacy concerns can be discovered through the ontology. This is an extremely important aspect of reasoning. Generally, most approaches to privacy assume that if an OSN is not instructed properly on how to treat a particular type of content, then the fault lies with the user. However, in many cases, OSN can, and should, make realistic assumptions based on what the user has already told it. For example, if a user did not want her e-mail address to be shown but has not made any specific demands about her phone number, one can assume that this information should also be treated private since both an e-mail address and a phone number are similar types of information; i.e., enable users to be accessed through different means. However, if no claims have been made about the age information, not much can be inferred since an e-mail address and age are categorically different. To realize this reasoning, we compute the similarity of concepts in the ontology. To keep this simple, we assume that siblings of a concept are similar and non-siblings are not. While this simplification is enough to illustrate our purposes, in a real system one can easily

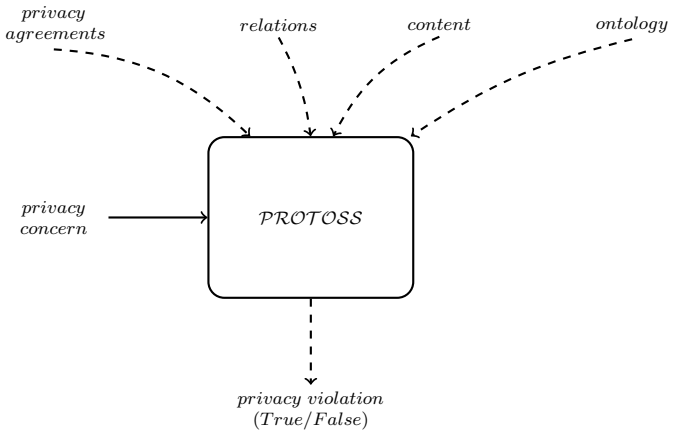

Fig. 5 Overview of the privacy violation detection process of $\mathcal{PROTOSS}$. $\mathcal{PROTOSS}$ (rectangle in the middle) takes a privacy concern (incoming straight line) along with various types of information from the social network (incoming dashed lines) as input and outputs whether the privacy concern holds (outgoing dashed lines).

apply more sophisticated semantic similarity metrics from the literature [20, 24] to compute more accurate similarities.

The ontology is used to make reasoning on the concepts, mainly to identify similar concepts. However, information on how concepts affect each other is kept separately as inference rules. An example inference rule is the following:

```
visible(location(Y, L), Z) ←
    visible(with(X, Y), Z) ∧
    visible(location(X, L), Z)
```

This rule states that if X and Y are together and this fact is visible to Z, then when Z knows the location of X, he will also know the location of Y. Another example is the following:

```
visible(picture(Y, I), Z) ←
    visible(picture(Y, I), X) ∧
    friend(X, Z)
```

This rule states that if picture I of Y is visible to X and X is a friend of Z, then Z can see I. The reasoning is that X may repost I and if so, due to friendship between X and Z, Z can see I. These rules define semantic relations among concepts in the real world so that $\mathcal{PROTOSS}$ can make further inferences beyond its privacy agreement.

Usage (Detection): The first use of $\mathcal{PROTOSS}$ is detecting privacy violations. Detecting a violation means that at the current state of the system, OSN violates at least one of its commitments to its users and therefore a user's privacy is at risk. OSN can check whether it is in such a situation by presenting privacy concerns that can be checked against the current system state. We present the overview of the privacy violation detection process in Fig. 5. The OSN operator feeds the privacy agreement, user relations, shared content and the ontology to $\mathcal{PROTOSS}$ in addition to a privacy concern and $\mathcal{PROTOSS}$ checks whether the concerned privacy breach exists in the current state of the OSN.

```
—P1
```

```
SPEC
 AG (colleague_charlie_linus ->
      AF c4.status = VIOLATED);

--P2
SPEC
 AG (colleague_charlie_linus ->
      AF c6.status = VIOLATED);
```

Above we give two example privacy concerns represented as CTL formulas. The first concern ($P1$) is about a possible breach of $charlie$'s privacy with respect to the visibility of his picture to his colleague $linus$ (i.e., C_4 is violated). Let us explain the privacy concern in detail. The privacy concern states that in all possible future states of the OSN (AG), if $charlie$ and $linus$ are colleagues (colleague_charlie_linus), then eventually C_4 is violated (AF c4.status = VIOLATED). $\mathcal{PROTOSS}$ uses the NuSMV model checker to check this privacy concern. Then, if the model checker returns true, which means the privacy concern holds, $\mathcal{PROTOSS}$ reports this privacy violation. On the other hand, if the model checker returns false, which means the privacy concern does not hold, $\mathcal{PROTOSS}$ reports $charlie$'s privacy is preserved. The second privacy concern ($P2$) is identical to first one except it is about the violation of commitment C_6.

Usage (Prediction): While checking privacy violations that have happened is important, predicting violations before they take place is maybe even more important. That is, if $charlie$ can predict that if he travels with $patty$ then his location will leak to his colleagues, then he might prefer not to travel or make an effort to hide his travels with $patty$. Hence, $charlie$ can actually avoid his personal information to reach third parties.

By its very nature, prediction is uncertain. In order to predict violations that might happen in a future state, one needs to make assumptions as to how the world will evolve. The prediction results simply state that if the world evolves in conformance with the assumptions, then a privacy violation will take place. In many cases, one can assess the likelihood of these assumptions. For example, assume $charlie$ predicts that, if $linus$ and $patty$ become friends, then his personal information will leak. Many time, $charlie$ will also have a fair idea as to $linus$ and $patty$ would become friends or not.

In order to realize the above reasoning, given a current state of the system, the user first makes some assumptions about a future state of the network. These assumptions are about the relations of other users and they are either true or false. Every other relation that is unknown to the system are set as free variables that can be instantiated to either true or false in different runs during the model checking process. With this state information, OSN checks for violations. Note that since some relations are set to unknown, not only this particular state is checked, but several variations of it are also checked. Fig. 6 shows the usage of $\mathcal{PROTOSS}$ for this purpose. Notice that, compared to Fig. 5, we additionally have a set of assumptions fed and the result is not only a decision on the violation but in the case of violation, a hypothetical scenario in which the violation would take place.

The privacy concern below (P_3) checks whether there is a chance that the OSN operator's commitment to $charlie$ (c4) is violated at some point after it has become active. Remember that this commitment states that $charlie$'s pictures will not be visible to $linus$ if they are not friends. A violation of this commitment means that $charlie$'s privacy may be jeopardized.

```
--P3
SPEC
AG (!friend_charlie_linus -> AF c4.status != VIOLATED)
```

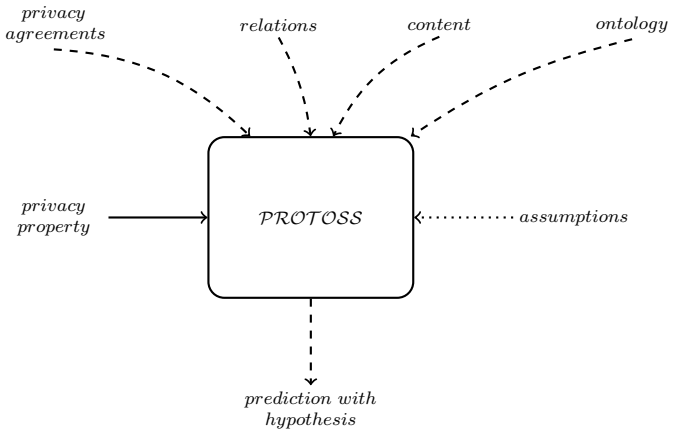

Fig. 6 Overview of the privacy violation prediction process of $\mathcal{PROTOSS}$. $\mathcal{PROTOSS}$ (rectangle in the middle) takes a privacy concern (incoming straight line) along with various types of information from the social network (incoming dashed lines) and a set of assumptions (incoming dotted line) as input and outputs a possible evaluation of the OSN where the privacy concern holds(outgoing dashed lines).

$\mathcal{PROTOSS}$ has been implemented in Java and uses NuSMV as a model checker[1]. It has two interfaces. The first interface is meant for detection and prediction. A system description file is loaded to the system and a privacy concern in the form of a CTL formula is given to be checked. For detection, the system file specifies all necessary relations as TRUE or FALSE and the checking of the privacy concern states whether there is a privacy violation or not. For prediction, the user can alter the relations, creating a prediction setting. The relations that are set as either TRUE or FALSE are known facts about the system. The relations that are left as TRUE, FALSE are those that can take any value in the future. By setting these relations as desired, a user can create a hypothetical future state and then ask OSN to check for violations according to the commitment given in the formula. The output shows the combination of relations that will lead to a violation, if any violation is to occur. Fig. 7 shows a screen-shot of this interface for prediction.

Usage (Analysis): If desired, $\mathcal{PROTOSS}$ can also be used to generate and analyze OSNs in terms of performance. When this is the case, one would want to generate an example OSN from scratch. In order to do this, we would need to feed information for generation of the OSN, such as the number of users, the relations, and so on. With this information, an example OSN can be generated and then studied. Fig. 8 shows a screen-shot of the second interface of $\mathcal{PROTOSS}$[2]. From the interface, we can create a social network (together with its relations) according to the number of users set or we can simply upload an existing social network specification, similar to the one in Section 4. If the creation option is set, then the relations and commitments are instantiated exhaustively. Once the model of the social network is created (on the left pane), we can again check whether the properties of interest are satisfied by the model. After the execution is completed, the output of the check (whether the privacy concern of interest holds) is shown with relevant performance statistics (on the right pane). These statistics contain information on the speed of checking as well as

[1] This implementation can be downloaded from `http://mas.cmpe.boun.edu.tr/ozgur/code.html`, under Section "5. Experiments for Predicting Privacy Violations with PROTOSS".

[2] This implementation can be downloaded from `http://mas.cmpe.boun.edu.tr/ozgur/code.html`, under Section "4. Experiments for Model Checking Privacy Agreements"

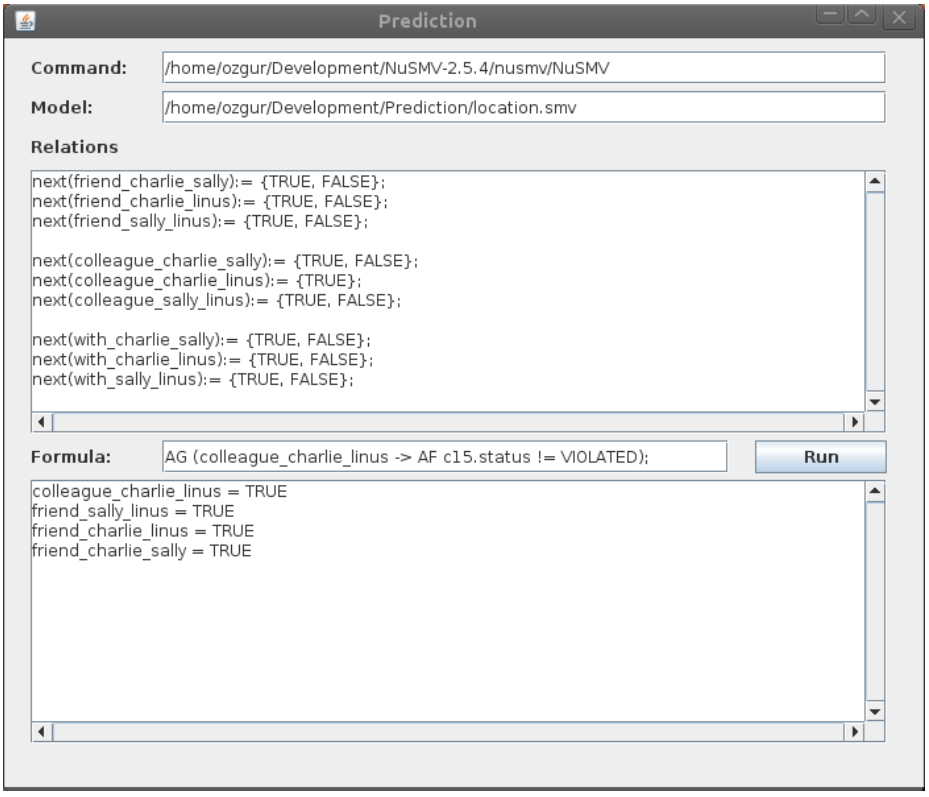

Fig. 7 $\mathcal{PROTOSS}$ interface for detecting and predicting privacy violations. In the case of prediction, the "Relations" panel is used to define assumptions. The lower panel shows the hypothesis that leads to a privacy violation (if any violation exists).

the number of states generated to perform the check. These can be useful in understanding the memory and speed requirements for various privacy concern checks for various OSN sizes.

We mainly evaluate our approach on the scenarios given in Sections 4. The aim of this evaluation is to depict how $\mathcal{PROTOSS}$ works on various cases. In addition to this, we provide a performance analysis to illustrate the resource requirements for the system.

5.1 Evaluations of Scenarios

Here, we simulate the scenarios described in Section 4. Then, based on the outcome produced by $\mathcal{PROTOSS}$, we comment on how these settings have an effect on the privacy of $charlie$. Table 1 shows the results of our execution for detecting privacy violations that correspond to Scenarios 1–5. False means that the privacy concern does not hold for that scenario, while true means the privacy concern holds. Since we are checking whether a commitment is violated, true corresponds to a violation and thus breach of privacy and false corresponds to no violation.

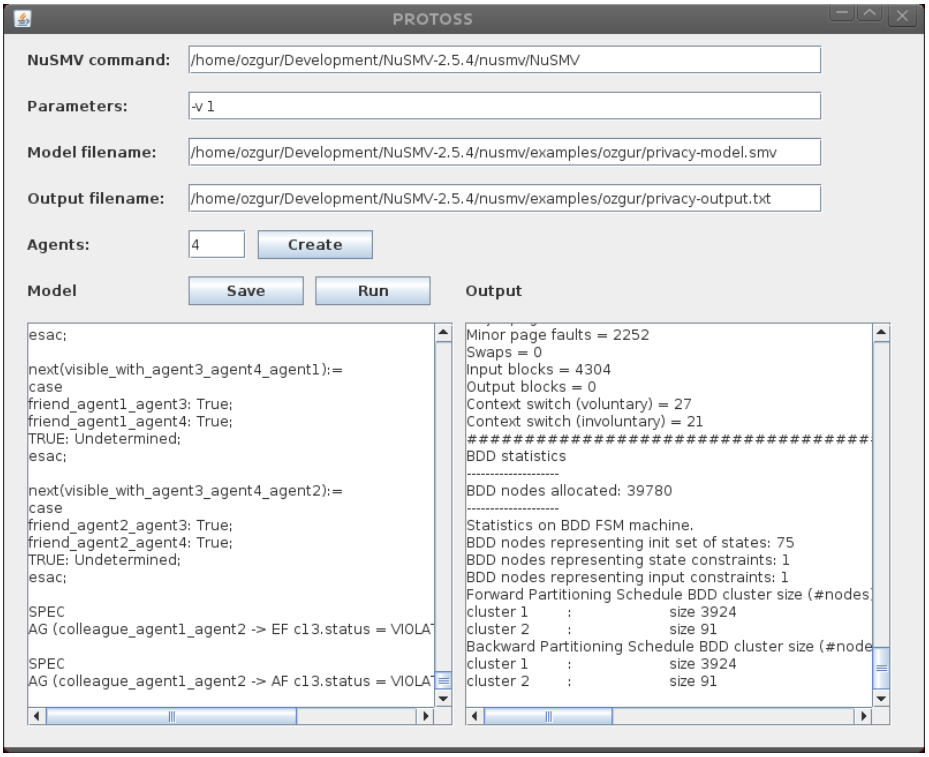

Fig. 8 $\mathcal{PROTOSS}$ interface for generating and analyzing social networks. Given the model parameters, the left panel shows the NuSMV model of the generated network. The right panel shows the result of the model checking process.

Scenario	Privacy Concern	Result
Scenario 1	AG(colleague_charlie_linus → AF c4.status = VIOLATED)	False
Scenario 2	AG(colleague_charlie_linus → AF c6.status = VIOLATED)	False
Scenario 3	AG(colleague_charlie_linus → AF c4.status = VIOLATED)	False
Scenario 4	AG(colleague_charlie_linus → AF c4.status = VIOLATED)	True
Scenario 5	AG(colleague_charlie_linus → AF c6.status = VIOLATED)	True

Table 1 Privacy violation detection results for the scenarios. The first column identifies the scenario, the second column shows the privacy concern and the third column presents the outcomes of the privacy violation detection.

Scenario 1: In this scenario, the OSN operator aims to detect whether $charlie$'s picture could be seen by his colleague $linus$ by violating his privacy agreement (i.e, C_4 is violated). Remember that the OSN is in its initial state as presented in Fig. 2. When we run $\mathcal{PROTOSS}$ for detection on this privacy concern we see that it returns false (Table 1). That is, C_4 is not violated and hence it is not the case that $charlie$'s picture is revealed to $linus$. Therefore, privacy of $charlie$ is preserved.

Scenario 2: In this scenario, the OSN operator aims to detect whether $charlie$'s location could be found by his colleague $linus$ by violating his privacy agreement (i.e, C_6 is vi-

olated). As in the previous scenario, the OSN is in its initial state as presented in Fig. 2. When we run $\mathcal{PROTOSS}$ for detection on this privacy concern we see that it returns false (Table 1). That is, C_6 is not violated and hence it is not the case that $charlie$'s location is revealed to $linus$. Therefore, privacy of $charlie$ is preserved.

Scenario 3: If we look at the privacy agreement of $charlie$ we see that there is no statement about $charlie$'s videos. However, as shown in Fig. 2, $charlie$ shares his video $beachVideo$. Since nothing is stated about videos in $charlie$'s privacy agreement, the OSN operator can show his videos to any other user without breaching $charlie$'s privacy. On the other hand, C_4 in the privacy agreement of $charlie$ states that $charlie$ does not want his pictures to be shown to his colleagues. Since both pictures and videos are semantically similar visual content the OSN operator may realize that $charlie$ might prefer to keep his videos not visible to his colleagues as his pictures. $\mathcal{PROTOSS}$ achieves this kind of reasoning using its semantic component. In this context, while checking the privacy concern about $charlie$'s pictures, $\mathcal{PROTOSS}$ also performs a similarity check among contents shared by $charlie$ that are similar to picture and discovers that $charlie$ shares a video (which is similar as a media type to a picture). From Scenario 1 we know that the pictures of $charlie$ are not shown to his colleagues, hence his privacy is preserved. Accordingly, we can conclude that $charlie$'s videos are also not visible to his colleagues. Without semantic reasoning, it would be difficult to make this conclusion. Instead a regular OSN operator would assume that since no commitment has been done about the videos, it would be free to distribute them. We can conclude that with semantic reasoning, OSN operator can detect this subtle similarity and can inform $charlie$ about the situation of his videos. In this case, $charlie$ would be free to update his privacy agreements to disable showing videos or to keep them the way they are now.

Scenario 4: In this scenario the OSN is evolved into the state in Fig. 3 due to the new relation between $charlie$ and $sally$. In this scenario, the privacy concern is whether $charlie$'s picture could be seen by his colleague $linus$ (i.e, C_4 is violated). When we run $\mathcal{PROTOSS}$ for detection on this privacy concern we see that it returns true (Table 1). That is, C_4 is violated and hence it is the case that $charlie$'s picture is seen by $linus$. The reason is the new friend relation between $charlie$ and $sally$. Due to this relation and the behavior rule B_3 of the OSN operator, $sally$ can see pictures of $charlie$. Besides, $sally$ can share these pictures too, which are visible to $linus$. Hence, $linus$ can see pictures of $charlie$ which violates c_3 and causes violation of the privacy agreement of $charlie$. Note that, in this scenario, the OSN operator does not reveal the picture of $charlie$ directly to $linus$, however, the picture is propagated to $linus$ to the chain of relationships between the users. Hence, such a violation cannot be captured, if the OSN operator only considers whether it reveals the picture of $charlie$ directly to $linus$.

Scenario 5: As in the previous case, in this scenario we consider the OSN is evolved into the state in Fig. 3 due to the new relation between $charlie$ and $sally$. In this scenario, the privacy concern is whether $charlie$'s location could be found by his colleague $linus$ (i.e, C_5 is violated). When we run $\mathcal{PROTOSS}$ for detection on this privacy concern we see that it returns true (Table 1). That is, C_5 is violated and hence it is the case that $charlie$'s location is found by $linus$. However, note that $charlie$ actually does not share his location to anybody. Hence, it is not expected $linus$ to find out his location. But this is a faulty expectation. Although, the location of $charlie$ is not shared by himself, his new friend $sally$ does this indirectly by telling that she is with $charlie$ and she is in Istanbul. Since $linus$ is a friend of $sally$, he can access to this information and easily conclude that $charlie$ is in Istanbul too. Note that, similar to the previous case, the OSN operator actually does not reveal the

location of $charlie$ directly to $linus$. Moreover, $charlie$'s location is actually not known by the OSN operator, since $charlie$ does not share this content. However, this information is deduced by $linus$ using the content shared by $sally$, which causes a privacy breach for $charlie$.

Next we study how $\mathcal{PROTOSS}$ responds to prediction cases. Table 2 summarizes our results for Scenarios 6–7. For each scenario, we show the privacy concern we check, its prediction result, and in case of violation an example evolution of the OSN in which the violation would take place. Remember that in all these scenarios the OSN is in its initial state (i.e., $charlie$ and $sally$ are not friends yet) and $charlie$ uses his own knowledge as we present in Fig. 4 and some assumptions about the relations of other users.

Experiment	Privacy Concern	Result	Hypothesis
Scenario 6	AG(!friend_charlie_linus → AF c4.status != VIOLATED)	FALSE	friend(sally,linus); friend(charlie,sally)
Scenario 7	AG(!friend_charlie_linus → AF c4.status != VIOLATED)	FALSE	friend(charlie,sally)

Table 2 Privacy violation prediction results for Scenarios 6–7. The first and second columns identify the considered scenario and privacy concern, respectively. The third column presents the outcome of the privacy violation prediction. The last column presents the hypothesis of the predicted violation (if one exists).

Scenario 6: In this scenario $charlie$ wants to predict what would it take for $linus$ to see his pictures (i.e., C_4 is violated)? In this case, $charlie$ does not make any assumption about the relations between the other users, since this information is usually not available. So, the only relations that $charlie$ is certain about are his friend relation with $patty$ and colleague relation with $linus$. Therefore, the other relations that are unknown in the OSN are set as free variables and $\mathcal{PROTOSS}$ considers them either existing or not-existing in different possible evolutions of the OSN relations. When we run $\mathcal{PROTOSS}$ for prediction in this setting, it returns false. That means if some new relations are initiated in the OSN, then C_4 will be violated and $linus$ can see $charlie$'s pictures. Besides capturing this situation, $\mathcal{PROTOSS}$ also generates a possible evolution of relations in the OSN that will lead to the breach of $charlie$'s privacy. According to the result of $\mathcal{PROTOSS}$, if $sally$ and $linus$ are friends, and $charlie$ initiates a friend relation with $sally$, then $charlie$'s pictures will be visible to $linus$. This corresponds to the setting given in scenario 3. However, note that here $\mathcal{PROTOSS}$ rather predicts these hypothetical relations (i.e., $friend(linus, sally)$ and $friend(charlie, sally)$) as a set of possible relations that might hold in a future state and thereby cause a violation. Also note that, this is not the only way for $linus$ to see $charlie$'s picture. For instance, if $linus$ and $patty$ also get friends, $linus$ can see $charlie$'s picture, even if $charlie$ and $sally$ are not friends. Hence, the possible evolution generated by $\mathcal{PROTOSS}$ is not necessarily be the only way of the OSN's evolution that causes a privacy violation. However, $\mathcal{PROTOSS}$ is sound and complete thanks to its model checking based approach. Hence, if there is such an evolution of relations for the OSN then it finds this evolution and if an evolution is found then it leads to a violation for certain.

Scenario 7: In Scenarios 5 $charlie$ uses only its own known relations without making any assumptions about the relations of the other users. However, in some situations, such information may be available from external sources. For instance, since $patty$ is a friend of $charlie$, he may know that she is not a friend of $linus$. In such a situation $charlie$ can feed this information to $\mathcal{PROTOSS}$ as an assumption and $\mathcal{PROTOSS}$ can use it to refine

its prediction process by reducing the number of free-variables due to unknown relations between other users. Generally speaking, when there are fewer free-variables, we expect $\mathcal{PROTOSS}$ to generate more accurate predictions. That is, if one knew all the relations, the result generated by $\mathcal{PROTOSS}$ would correspond to detection; i.e., the actual state of the system. When we run $\mathcal{PROTOSS}$ for prediction in this setting, it returns false. That means if some new relations are initiated in the OSN, then C_4 will be violated and $linus$ can see $charlie$'s pictures. However, this time $\mathcal{PROTOSS}$ does not generate the possible evolution of relation that state a friend relation between $patty$ and $linus$ leads to violation of C_4, since $charlie$ explicitly states $patty$ and $linus$ are not friends. Hence, $linus$ sees $charlie$'s picture only if $charlie$ and $sally$ become friends.

The above prediction examples focus on the pictures as content, however similar predictions can be made when the content being shared is a video or a location.

5.2 Performance Results

Now, we briefly study the performance results related to the workings of $\mathcal{PROTOSS}$ on our privacy models. Table 3 shows the performance of $\mathcal{PROTOSS}$ for variations of Scenario 1 on an Intel Core i7 2.9 GHz computer with 6 GB of memory running Ubuntu 12.04 64-bit OS. While the main setting is the same, we generate OSNs with varying sizes of users. In our generations, all the users are related to each other through relations so during checking privacy concerns we take the entire OSN into account. The results demonstrate the number of states, the memory used, and the time consumed based on 3 to 20 user models. Note that the time and memory consumption increases exponentially since the model grows based on the number of users, i.e., the number of possible relations among the users increase exponentially with the number of users. For small networks, the computation times lie within reasonable amounts. However, with large networks, the checking time can become intolerable for quick decisions.

We have also collected the performance statistics for the prediction Scenarios (6–7). Table 4 shows the performance of $\mathcal{PROTOSS}$ for those scenarios in terms of the number of states, the memory size, and the speed. Remember that for Scenario 6, we had stated that the user would not make assumptions about how the relations would evolve. Hence, we had left most of the relations as free variables to take any possible value. This means that $\mathcal{PROTOSS}$ would need to try different variations to see if anyone of those would violate the privacy agreement. On the other hand, for Scenario 7, the user would make some assumptions about how the relations would evolve and set them accordingly. Hence, there were fewer cases to test for $\mathcal{PROTOSS}$. This difference is also reflected in the performances as seen in Table 4. The first case requires longer time to generate an answer compared to the last cases. The memory requirements are harder to analyze. We speculate that different state representations require different memory sizes, resulting in variations on the total memory size. Analysis of memory as well as improvements on the speed are both important directions for our future work.

5.3 Experimenting with Real Data

In order to understand how our tool can be applied to real-life social networks, we experiment with an existing Facebook dataset [23]. This dataset consists of rows, where each row lists two individuals that are related to each other and optionally a date that implies

#Users	#States	Time
3	793	0.04 s
4	3.5 K	0.06 s
5	11.6 K	0.08 s
6	31.9 K	0.09 s
7	67.7 K	0.11 s
8	117.1 K	0.13 s
9	137.4 K	0.16 s
10	217.4 K	0.18 s
11	299.8 K	0.25 s
12	435.4 K	0.31 s
13	672.8 K	0.45 s
14	1.1 M	0.71 s
15	1.5 M	2.2 s
16	3.1 M	3.1 s
17	6.3 M	5.9 s
18	13.4 M	12.5 s
19	27.6 M	29.2 s
20	57.3 M	68.5 s

Table 3 Performance results for variations of Scenario 1. The columns #Users, #States and Time show the number of users used for evaluation, the number of states generated, and the time spent for generating an answer, respectively.

Scenario	#States	Time
Scenario 6	13.9 K	0.812 s
Scenario 7	14.1 K	0.734 s

Table 4 Performance results of prediction scenarios. The columns Scenario, #States and Time denote the scenario number, the number of states generated, and the time spent for model checking, respectively.

when the relationship between the two individuals were formed. This dataset is much simpler compared to our setup above. Specifically, it does not contain different type of relations or contents, explicit privacy policies (with or without commitments), or any system behavior rules. With this dataset, we create a fictitious but realistic setting. We assume that the relations between the two individuals are friend relations. If the date is available, we take that as the date the relationship was formed. If the date is not available, we assume that the relationship was there all along. As is the usual privacy agreement in many social networks, we assume that OSN will show the content posted by a user (e.g., pictures) to her friends, but not anyone else. We reflect this in the privacy agreement of the user by a commitment $C(osn, user, \neg\, friend(user, X), \neg\, visible(picture(user), X))$. Again, as is usual in many online social network systems, we assume that a user can repost a content that was initially posted by a friend.

In this setting, we focus on one particular type of privacy violation. Is it possible for a user Y to actually view a content posted by X, even though X and Y are not friends? If so, can we predict it before it happens? In order to answer these questions, we take a subset of the dataset such that we begin with one user and add all of her friends and her friends'

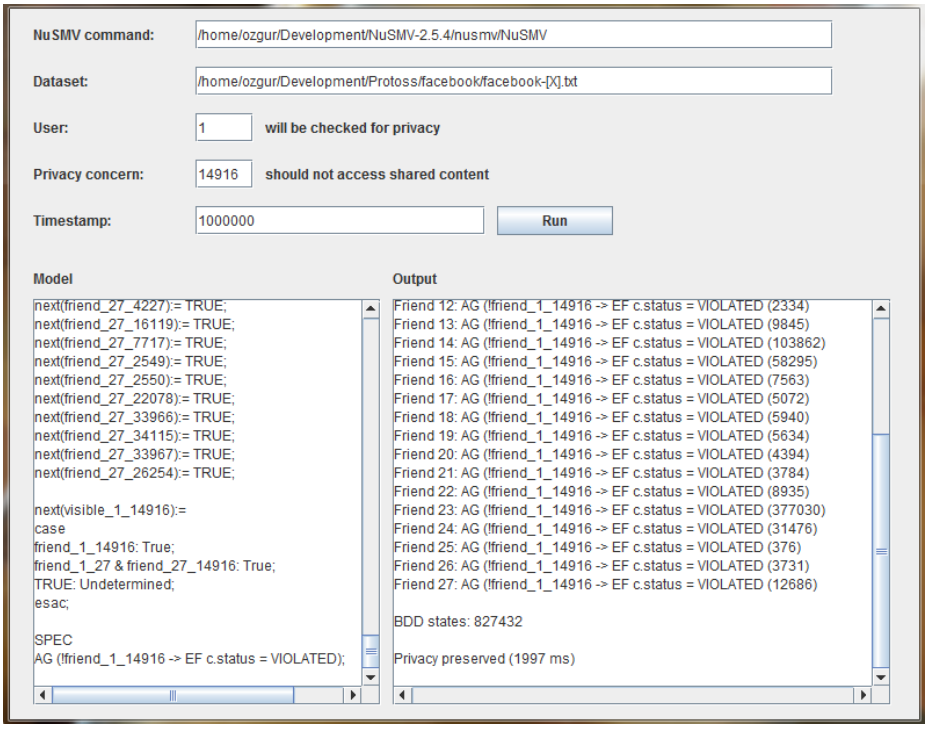

Fig. 9 $\mathcal{PROTOSS}$ interface for the Facebook dataset.

friends[3]. Then we partition the dataset into two based on a given date information. The idea is to have one set of relations that already exist and a second set that will be formed in the future according to this partition. We envision ourselves as answering questions at a time after the first set is formed but before the second set of relations is formed. This way, we can make predictions and then check whether those predictions hold or can hold using the second partition.

Going back to the question, we can actually find a scenario where Y ends up viewing a content of X, because there exists a Z that is both friends with X and Y, and shares the content of X with Y. Hence, OSN's commitment to X is violated. Note that this setting is similar to Scenarios 1 and 3. Here, we can predict the violation of the commitment using $\mathcal{PROTOSS}$ if we check at the right time, meaning before Y and Z become friends. Fig. 9 shows the interface of $\mathcal{PROTOSS}$ that realizes this setting. The input is the user to be checked for privacy (X), the privacy concern of the user (Y will not view her content), and the timestamp to partition the dataset. Once $\mathcal{PROTOSS}$ is run with this input, it will create a NuSMV model for each friend Z of X, and check the privacy agreement of X with friends of Z.

The output for a sample user 1645 from the dataset is shown in the figure. She does not want user 821 to view her content given that they are not friends. $\mathcal{PROTOSS}$ successfully predicts that her privacy would be violated if 821 and one of 1645's friends were also friends.

[3] In order to have a performance gain, we stop expanding the network at the friends of friends level. Recent work on link prediction in social networks [3] have shown that it is very likely for a new friend to be already contained in this friends of friends network.

It can be seen from the figure that this is the case for users 16 and 53. With this information, OSN can either warn 1645 if 821 and 16 attempt to become friends or may not allow this relation to be formed in the first place.

We have also tested the performance of $\mathcal{PROTOSS}$ on the Facebook dataset using the same computer specifications as in Section 5.2. We have made five sets of experiments by selecting random users from the dataset. In each set, the user checks five different privacy concerns. The details are summarized in Table 5, where for each user we list the number of immediate friends, the total number of friend relations in the network, average number of friends per user, and the prediction time. The results are reasonable with prediction times of 1.19 seconds for a network of 45 friend relations for each user on average. There is a linear correlation between the number of friend relations and prediction times.

User	#Users	#Friends	#States	Prediction time
1	27	2129	894.4 K	1.75 s
163	26	1222	396.2 K	0.94 s
1645	29	679	127.1 K	0.89 s
31720	50	2294	557.6 K	1.87 s
48696	16	495	144.1 K	0.50 s

Table 5 Number of friends vs. prediction time in the Facebook dataset. #Users shows the number of users in the network for which we have included the friend links (i.e., the number of friends the user has plus the user herself). Note that the actual number of users in the network is much larger. #Friends shows the total number of links between the users. #States shows the total state space used by the model checker. The average prediction times are recorded in seconds.

To improve prediction times, we've pruned the model for each verification run appropriately so that the state space does not increase exponentially as in Section 5.2. We've built a NuSMV model for each friend of the user and run it separately. This way, we could avoid a big model that would lead to a state explosion. In addition, we do not take into account all the relations among other users of the network which will not contribute to the privacy check (thanks to single friend relation in the Facebook dataset). You can see from Table 5 that even though we have more users than the number of users in the experiments for Section 5.2, our model stays within reasonable state spaces.

6 Discussion

Preserving privacy of users is an essential part of any Web system, including OSNs where information can travel fast and far. We next review relevant literature and then point out directions for future work.

Related Work: Krishnamurthy and Wills study the leakage of personal identifiable information in social networks [16]. They consider personal identifiable information as any piece of information that can by itself or when combined with other information help decipher a person's identity. They depict different ways that such information can leak to external applications. The types of leakages they are concerned with are generally based on HTTP side effects that allow information to appear in URLs or cookies that can be used by other applications. While those identified leakages by the authors are important, the types of leakages we are concerned here are more high-level and may still exist even if the leakages due to HTTP side effects are handled

Fang and LeFevre point out that following a privacy agreement for a novice user is difficult and there is a need for easy-to-use privacy specification tools for such users [12]. To address this, they develop a learning-based privacy wizard that asks for some example cases from a user and learns with whom a user wants to share information and what kind of. Based on this learning, the wizard decides on the privacy settings of the user. This is certainly an important aspect of privacy. In our work, we assume that privacy agreements can be represented and processed formally so that we can focus on the interplay between privacy agreements and user relations.

Akcora, Carminati, and Ferrari measure how risky an individual in a social network is in terms of privacy [2]. They develop a method in which a user's friends' friends are analyzed in terms of their potential for learning and misusing personal information. The approach is based on active learning and hence the risk is decided by the community itself. They have adapted their approach to Facebook to detect people that can potentially violate privacy of a user. While their aim is to identify risky individuals, our aim in this work is to help a system decide on potential privacy violations that would stem from the relations in the system and inform the user appropriately.

Xue, Carminati, and Ferrari develop a protocol for identifying privacy-preserving paths in social networks [25]. They view the social network as a graph, where nodes are the users and the edges are labeled with the relations among users. On this graph, they find paths from one user to another that respects the relation type, strength, and trust level among users. While this problem is different from the one we address in this paper, it would be interesting to analyze the paths found in this approach and compare their relations to the cases in which we detect privacy violations.

There is a large body of research on anonymization of data, including data extracted from online social networks. Li, Zhang, and Das thoughtfully organize different techniques that preserve privacy of social relations [18]. The general problem there is the deciphering of social relations by attackers even after the data has been anonymized at different extents. The identification of such relations lead to privacy violations not only for the users from whom the information was extracted in the first place but for those who reside on the other side of the relation.

Carminati *et al.* study access controls in online social networks through semantic Web technologies, such as OWL and SWRL [6]. They model the semantic relations among users, the resources, and actions through these technologies. They then execute security policies using these semantic information. Our use of ontologies in this work carries a similar flavor in representing and reasoning on the actions of users. While their focus is on access control, our focus is on commitments and their violations. Hence, they use their system to decide on accesses, while we use it to detect undesirable conflicts among policies and relations. We also use semantic similarities between contents to infer privacy expectations of users.

Model checking has been used to identify a variety of failures in commitment-based systems. Bentahar *et al.* consider social commitments and its operations in a Kripke-like world model [5]. The commitment semantics and related operations are formalized with CTL using that world model, and some properties are justified regarding desired execution of commitment protocols. Using model checking, they verify the compliance of agents to their commitments. In [10], two properties of commitments are considered for verification; fulfillment and violation. In addition, safety and liveness properties of commitment protocols are verified. In [22], Telang and Singh model several business patterns as commitment interactions and map them onto CTL specifications. Then, using model checking they verify whether the underlying operational model (built with commitment semantics and its operations) supports the business specifications. There are major differences between these works

and our work. First of all, we are not only interested in checking whether commitments have been violated or fulfilled through agents' actions but also through the relations that bind the agents. This means that even if an agent does not violate a commitment through explicit actions, its relations with other agents can make the commitment impossible to discharge. Second, both in the works of Bentahar *et al.* and Telang and Singh, the aim is to analyze the system at design time. This is helpful to see if any commitments are going to be violated. However, in our work relations are created at run time. Hence, we apply the model checking at run time.

Parallel to the increasing popularity of OSNs various models have been developed to capture evolutionary dynamics of OSNs [17, 23]. These studies explain how the relations between users in various OSNs evolve. A major finding of these studies is that most new relations are formed between individuals, who are connected to each other through acquaintances. In this context, a very frequent behavior of users is closing triangles. For instance, if the user $charlie$ is a friend of users $patty$ and $sally$, then it is frequently observed that $patty$ and $sally$ also become friends at a future point. Integration of these kind of models into our approach is a promising direction to improve both our detection and prediction performances. For instance, while predicting possible privacy violations of $charlie$, instead of considering all the users and their possible future relations, our approach may consider only users that are close to $charlie$ in the OSN, since in the light of these observations it is reasonable to assume new relations that may cause to privacy violations are going to be formed only between these close users.

Conclusion: We demonstrate here that privacy violations take place not only because OSN operators are not meticulous enough, but also from the way an OSN system accommodates various relations. $\mathcal{PROTOSS}$ uses commitments and model checking to detect privacy violations in OSNs. In various situations, it can perform semantic reasoning on its existing knowledge to signal potentially risky situations, which were not initially identified as violations by the user. An important aspect of $\mathcal{PROTOSS}$ is that it can also predict violations before they take place. Hence, it can tell its users that if certain relations or facts take place in the world, then their privacy will be jeopardized. Users can then take necessary actions to avoid violations.

In our future work, we plan to integrate $\mathcal{PROTOSS}$ into an existing OSN. The Facebook data that we experimented with was a useful first step. However, for a large scale experiment we would need to integrate our agreement structure and behavior rules with an existing OSN. Such an integration will enable users to take the results of $\mathcal{PROTOSS}$ evaluation into account when deciding to upload content. One major challenge for this integration is the performance. We plan to apply reduction techniques (e.g., use of relations instead of individuals) to diminish the state space for model checking so that we can compute with a large number of relations and inference rules. The second challenge is to develop a realistic ontology on which one can specify realistic relations between concepts and compute accurate similarities so that unspecified privacy expectations of the user can be computed more accurately.

Acknowledgment

We are indebted to Alan Mislove for sharing the Facebook dataset. This research is supported by Bogazici University Research Fund under grant BAP5694, and the Turkish State Planning Organization (DPT) under the TAM Project, number 2007K120610. Akın Günay is partially supported by a TÜBİTAK Scholarship (2211). Pınar Yolum is partially supported

by a TÜBİTAK Scholarship (2219). Most of this work was done while Özgür Kafalı was at Bogazici University, and Pınar Yolum was on sabbatical at Cornell University.

References

1. Lane v. facebook, inc. Wikipedia entry. Available at: http://en.wikipedia.org/wiki/Lane_v._Facebook,_Inc.
2. Akcora, C.G., Carminati, B., Ferrari, E.: Privacy in social networks: How risky is your social graph? In: Proceedings of the 28th International Conference on Data Engineering (ICDE) (2012). To appear.
3. Backstrom, L., Leskovec, J.: Supervised random walks: predicting and recommending links in social networks. In: Proceedings of the fourth ACM international conference on Web search and data mining, WSDM '11, pp. 635–644. ACM, New York, NY, USA (2011)
4. Baden, R., Bender, A., Spring, N., Bhattacharjee, B., Starin, D.: Persona: An online social network with user-defined privacy. In: Proceedings of the ACM SIGCOMM 2009 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM), pp. 135–146 (2009)
5. Bentahar, J., El-Menshawy, M., Dssouli, R.: An Integrated Semantics of Social Commitments and Associated Operations. In: Proceedings of the Second Multi-Agent Logics, Languages, and Organisations Federated Workshops, pp. (2009)
6. Carminati, B., Ferrari, E.: Privacy-aware access control in social networks: Issues and solutions. In: Privacy and Anonymity in Information Management Systems, chap. 9, pp. 181–195. Springer Verlag (2010)
7. Cimatti, A., Clarke, E., Giunchiglia, E., Giunchiglia, F., Pistore, M., Roveri, M., Sebastiani, R., Tacchella, A.: NuSMV Version 2: An OpenSource Tool for Symbolic Model Checking. In: Proc. International Conference on Computer-Aided Verification (CAV 2002), *LNCS*, vol. 2404. Springer, Copenhagen, Denmark (2002)
8. Clarke, E.M., Emerson, E.A.: Design and Synthesis of Synchronization Skeletons Using Branching-Time Temporal Logic. In: Logic of Programs, Workshop, pp. 52–71. Springer-Verlag, London, UK (1982)
9. Debatin, B., Lovejoy, J.P., Horn, A.K., Hughes, B.N.: Facebook and online privacy: Attitudes, behaviors, and unintended consequences. Journal of Computer-Mediated Communication **15**(1), 83–108 (2009)
10. El Menshawy, M., Benthar, J., Qu, H., Dssouli, R.: On the verification of social commitments and time. In: Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS), pp. 483–490 (2011)
11. Emerson, E.A.: Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B), chap. Temporal and Modal Logic, pp. 995–1072. MIT Press, Cambridge, MA, USA (1990)
12. Fang, L., LeFevre, K.: Privacy wizards for social networking sites. In: Proceedings of the 19th International Conference on World Wide Web (WWW), pp. 351–360 (2010)
13. Gruber, T.R.: A translation approach to portable ontology specifications. Knowledge Acqusition **5**(2), 199–220 (1993)
14. Heussner, K.M.: Celebrities' photos, videos may reveal location. ABC News. Available at: http://abcnews.go.com/Technology/celebrity-stalking-online-photos-give-location/story?id=11162352
15. Huth, M., Ryan, M.: Logic in Computer Science: Modelling and Reasoning About Systems, 2nd edn. Cambridge University Press (2004)
16. Krishnamurthy, B., Wills, C.E.: On the leakage of personally identifiable information via online social networks. Computer Communication Review **40**(1), 112–117 (2010)
17. Leskovec, J., Backstrom, L., Kumar, R., Tomkins, A.: Microscopic evolution of social networks. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 462–470. ACM, New York, NY, USA (2008)
18. Li, N., Zhang, N., Das, S.K.: Preserving relation privacy in online social network data. IEEE Internet Computing **15**(3), 35–42 (2011)
19. McGuinness, D.L.: Ontologies come of age. In: Spinning the Semantic Web, pp. 171–194. MIT Press, Cambridge (2003)
20. Resnik, P.: Semantic Similarity in a Taxonomy: An Information-based Measure and its Application to Problems of Ambiguity in Natural Language. Journal of Artificial Intelligence Research **11**, 95–130 (1999)
21. Singh, M.P.: An ontology for commitments in multiagent systems: Toward a unification of normative concepts. Artificial Intelligence and Law **7**, 97–113 (1999)

22. Telang, P., Singh, M.: Specifying and verifying cross-organizational business models: An agent-oriented approach. IEEE Transactions on Services Computing **4** (2011)
23. Viswanath, B., Mislove, A., Cha, M., Gummadi, K.P.: On the evolution of user interaction in facebook. In: Proceedings of the 2nd ACM SIGCOMM Workshop on Social Networks (WOSN'09) (2009)
24. Wu, Z., Palmer, M.: Verbs Semantics and Lexical Selection. In: Proceedings of the 32nd Annual Meeting on Association for Computational Linguistics (ACL), pp. 133–138 (1994)
25. Xue, M., Carminati, B., Ferrari, E.: P3d - privacy-preserving path discovery in decentralized online social networks. In: Proceedings of the 35th Annual IEEE International Computer Software and Applications Conference (COMPSAC), pp. 48–57. IEEE Computer Society (2011)
26. Yolum, P., Singh, M.P.: Flexible Protocol Specification and Execution: Applying Event Calculus Planning using Commitments. In: Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS), pp. 527–534 (2002)