A Logical Approach to Restricting Access in Online Social Networks

Marcos Cramer
University of Luxembourg

Jun Pang
University of Luxembourg

Yang Zhang
University of Luxembourg

ABSTRACT

Nowadays in popular online social networks users can blacklist some of their friends in order to disallow them to access resources that other non-blacklisted friends may access. We identify three independent binary decisions to utilize users' blacklists in access control policies, resulting into eight access restrictions. We formally define these restrictions in a hybrid logic for relationship-based access control, and provide syntactical transformations to rewrite a hybrid logic access control formula when fixing an access restriction. This enables a flexible and user-friendly approach for restricting access in social networks. We develop efficient algorithms for enforcing a subset of access control policies with restrictions. The effectiveness of the access restrictions and the efficiency of our algorithms are evaluated on a Facebook dataset.

Categories and Subject Descriptors

K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

Keywords

Online social networks; hybrid logic; access control; blacklist

1. INTRODUCTION

Online social networks (OSNs) have been the dominating applications in the Internet during the past few years. Leading actors, including Facebook, Twitter and Instagram, have a large number of users. Facebook has more than one billion monthly active users, 500 million tweets are published on Twitter everyday, and Instagram users share more than 70 million photos and videos on a daily base. Nowadays, OSNs have become an indispensable part of people's life.

A user can perform a lot of activities in OSNs, such as building his profile, articulating social relationships and publishing photos and statuses. In addition, OSNs have provided access control schemes for users to decide who can view their resources. The access control schemes in OSNs are relationship-based. In simple terms, a user can define access control policies to allow others who are in a certain relationship with him to access his resources.

SACMAT'15, June1–3, 2015, Vienna, Austria.

ACM 978-1-4503-3556-0/15/06 ...$15.00.
DOI: http://dx.doi.org/10.1145/2752952.2752967.

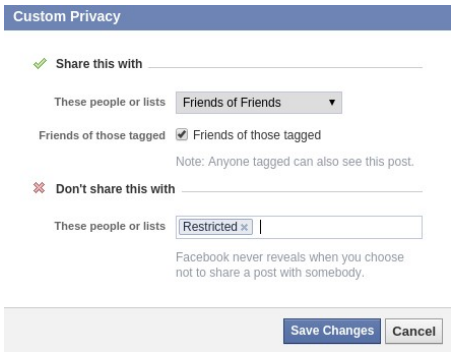


Figure 1: Access control with blacklist in Facebook

With more personal information published in OSNs, sometimes a user can be bothered by others, e.g., due to harassment or different political views. To deal with this, major OSN companies have provided functionalities to allow a user to put someone on his *blacklist.*[1] Those who are on a user's blacklist are still his friends but they are forbidden automatically to access his resources. For example, in Facebook, if a user only allows his friends to view his profile, then friends on his blacklist are disallowed to access his profile directly.[2] In this way, blacklists can be treated as orthogonal to access control policies. Figure 1 shows that a Facebook user can define a policy to share his post with his friends of friends but not with those on his blacklist.

However, to the best of our knowledge, the use of blacklists for restricting access in OSNs has not been well-understood and formally studied. For instance, suppose Alice and Bob are friends and Charlie is on Bob's blacklist. If Alice wants to share her photo with her friends of friends, should she *also* consider Bob's blacklist to deny Charlie's access? To address such research problems, we propose a logical approach to formalizing blacklist and its utilization in access control policies.

[1]It is called *Restricted list* in Facebook and *list of muted accounts* in Twitter.
[2]Note that adding someone into a blacklist is different from blocking him. This later is referred as *unfriending* in Facebook and *unfollowing* in Twitter, while blacklists do not change any relationship.

Our contributions. We summarize our main contributions in this paper as follows.

- We adopt a hybrid logic [12, 4] to specify access control policies in OSNs (Section 2). In order to better describe blacklist in policies, we propose a new path semantics for the logic and prove that it is equivalent to the original semantics of the logic (Section 3).
- Depending on different requirements, we classify three dimensions on how blacklists can be considered, namely *globality*, *generality* and *strength*. Each dimension is a binary decision, giving rise to eight flexible restrictions for users to use blacklists in their policies (Section 4).
- Since each policy can be affiliated with eight different blacklist-restrictions, in order to free users from the burden of defining access control policies precisely and correctly, we propose a syntactical transformation to rewrite an access control formula into its corresponding formula under a blacklist-restriction. In this way, a user only needs to define a policy and a restriction, our transformation will then generate the corresponding formula for enforcement automatically (Section 5).
- Most access control policies in OSNs mainly concentrate on the length of the path between the owner and the requester. Therefore, to improve the evaluation efficiency of this type of policies, we develop new algorithms for finding paths between the owner and the requester under different blacklist-restrictions. Experiments on a real-life social network dataset demonstrate their efficiency (Section 6 and Section 7).
- We perform experiments to study the effect of blacklist-restrictions on access control policies. We find that the restriction from the *strength* dimension is more powerful than from the other two dimensions. In order for a requester to access the owner's information, we also find that he should have different social closeness to the owner for different blacklist-restrictions (Section 7).

2. A HYBRID LOGIC

In this section we present the hybrid logic introduced in [12, 4] for relationship-based access control in OSNs.

An online social network (OSN) is modeled as a directed graph, called *social graph*, and is denoted by $\mathcal{G} = (\mathcal{U}, \mathcal{E})$, where the set $\mathcal{U}$ of nodes consists of the users in the OSN, and the set $\mathcal{E}$ of labeled edges represents the relationships between the users. We use $\mathcal{R} = \{\alpha_1, \ldots, \alpha_m\}$ to denote a (finite) set of relationship types supported in the OSN. The semantics of each relationship type can be defined as $\alpha_i \subseteq \mathcal{U} \times \mathcal{U}$. For two users $u, u' \in \mathcal{U}$, if they are in a relationship of $\alpha_i \in \mathcal{R}$, we say $(u, u') \in \alpha_i$. Moreover, each user is affiliated with some basic information which are treated as attributes of the user. Figure 2 depicts a sample social graph.

For every resource, the owner of the resource can specify an access control policy for determining which users have access to the resource. In the logic [12, 4], we have two distinguished variables own and req for referring to the owner of the resource in question and the user requesting access.

Syntax. The syntax of the hybrid logic is given below.

$$t ::= n \mid x$$
$$\phi ::= t \mid p \mid \neg\phi \mid (\phi_1 \wedge \phi_2) \mid (\phi_1 \vee \phi_2) \mid \langle\alpha_i\rangle\phi \mid @_t\phi \mid \downarrow_x \phi$$

In order to explain the meaning of the symbols and operators informally, we first need to point out that a formula is always evaluated at some user in the graph. The logic supports two kinds of atoms, namely nominals (n) that represent a user's name in the social graph, and variables(x). Terms can function as formulas; they express that the user at which the formula is being evaluated is identical to the user referred to by the term. Propositional symbols (p) are used for representing attributes of the user at which they are evaluated. Negation ($\neg$), conjunction ($\wedge$) and disjunction ($\vee$) have their usual meanings. The intended meaning of the modal operator $\langle\alpha_i\rangle\phi$ is that $\langle\alpha_i\rangle\phi$ is true at a user u iff ϕ is true at some user u' such that u and u' stand in relationship α_i. The hybrid logic operator $@_t$ specifies that the formula following it should be evaluated at the user that the term t refers to. $\downarrow_x$ assigns the user at which the formula is evaluated to the variable x.

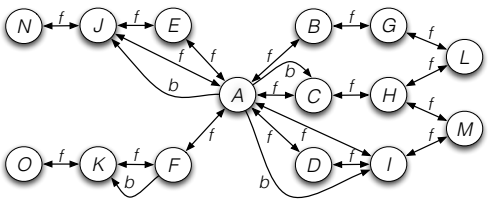


Figure 2: A social graph example

The set of formulas of the hybrid logic is denoted by L. We write $(\phi \rightarrow \psi)$ as an abbreviation for $(\neg\phi \vee \psi)$, and follow the usual conventions for dropping brackets in formulas.

Semantics. A *model* Γ is a triple $(\mathcal{G}, W, V)$, where $\mathcal{G}$ is a social graph, for every nominal n, $W(n)$ is a user in $\mathcal{G}$, and for every propositional variable p, $V(p)$ is a set of users that have the attribute as specified by p. A *valuation* is a map from variables to $\mathcal{U}$. The formulas of the logic are evaluated on triples Γ, u, τ, where $u \in \mathcal{U}$ and τ is a valuation:

$$\begin{array}{ll}
\Gamma, u, \tau \vDash x & \text{iff } u = \tau(x) \\
\Gamma, u, \tau \vDash n & \text{iff } u = W(n) \\
\Gamma, u, \tau \vDash p & \text{iff } u \in V(p) \\
\Gamma, u, \tau \vDash \neg\phi & \text{iff } \Gamma, u, \tau \nvDash \phi \\
\Gamma, u, \tau \vDash \phi_1 \wedge \phi_2 & \text{iff } \Gamma, u, \tau \vDash \phi_1 \wedge \Gamma, u, \tau \vDash \phi_2 \\
\Gamma, u, \tau \vDash \phi_1 \vee \phi_2 & \text{iff } \Gamma, u, \tau \vDash \phi_1 \vee \Gamma, u, \tau \vDash \phi_2 \\
\Gamma, u, \tau \vDash \langle\alpha_i\rangle\phi & \text{iff } \exists\, u' \in \mathcal{U} \text{ s.t. } (u, u') \in \alpha_i \wedge \Gamma, u', \tau \vDash \phi \\
\Gamma, u, \tau \vDash @_n\phi & \text{iff } \Gamma, u', \tau \vDash \phi, \text{ where } W(n) = u' \\
\Gamma, u, \tau \vDash @_x\phi & \text{iff } \Gamma, \tau(x), \tau \vDash \phi \\
\Gamma, u, \tau \vDash \downarrow_x \phi & \text{iff } \Gamma, u, \tau[x \mapsto u] \vDash \phi
\end{array}$$

Access control policies. The formulas in the hybrid logic are used to express an *access control policy* that specifies the conditions under which the access requester gets access to a resource depending on his relation to the owner of the resource. We define a subset of formulas of the hybrid logic which can be meaningfully applied for this purpose:

Definition 1. Let $L(\mathsf{own}, \mathsf{req})$ be the set of formulas of the hybrid logic that

- contain at most own and req as free variables, and
- are Boolean combinations of formulas of the two forms $@_{\mathsf{own}}\phi$ and $@_{\mathsf{req}}\phi$.

An element of $L(\mathsf{own}, \mathsf{req})$ is called an *access control policy*.

A user u can specify a policy ϕ for every resource he owns. For determining whether a user u' gets access to the re-

source, it needs to be checked whether $\Gamma, u, \tau \models \phi$. We use u_{own} to denote the owner and u_{req} to denote the requester. In the following discussions, we often refer to the policies $@_{\mathsf{own}}\langle f\rangle\langle f\rangle\mathsf{req}$ and $@_{\mathsf{own}}\langle f\rangle\langle f\rangle\langle f\rangle\mathsf{req}$. The first one expresses that u_{req} is a friend of a friend of u_{own}, we call this policy the *2-depth policy*. The second one expresses that u_{req} is three *friend* steps away from u_{own}, and is called the *3-depth policy*.

3. PATH SEMANTICS

In this section, we introduce a new definition of the semantics of the hybrid logic, which we call *path semantics*. It is equivalent to the standard semantics (see Theorem 1 below), but it allows us to refer to the set of paths in the social graph that makes a formula true. Being able to refer to this set of paths is important for defining the different ways in which blacklists can be used for restricting access.

When a formula is satisfied, there is a set of paths in the social graph that witnesses the truth of the formula. In Figure 2, taking A as the owner and M as the requester, the formula $@_{\mathsf{own}}(\langle f\rangle\langle f\rangle\mathsf{req} \wedge \langle f\rangle\langle f\rangle\langle f\rangle\mathsf{req})$ is satisfied, and this satisfaction is witnessed by the set $\{(A, I, M), (A, D, I, M)\}$ (path (A, I, M) witnesses $@_{\mathsf{own}}\langle f\rangle\langle f\rangle\mathsf{req}$, path (A, D, I, M) witnesses $@_{\mathsf{own}}\langle f\rangle\langle f\rangle\langle f\rangle\mathsf{req}$). This notion of a set of paths witnessing a formula can be formalized by defining the semantics of hybrid logic with reference to sets of paths.

For formalizing this new path semantics, we first define a path π to be a sequence of edges $\langle e_0, e_1, \ldots, e_n\rangle$, where $e_i \in \mathcal{E}$ for $0 \leq i \leq n$.[3] For such a path π, $\pi[k]$ denotes e_k, $\pi[1:]$ denotes $\langle e_1, \ldots, e_n\rangle$ and $e \circ \pi$ denotes $\langle e, e_0, e_1, \ldots, e_n\rangle$. For a set Π of paths, $\Pi[1:]$ denotes $\{\pi[1:] \mid \pi \in \Pi\}$.

The path semantics for the hybrid logic is given as follows:

$\Gamma, u, \Pi, \tau \models x$ iff $\Pi = \{\langle\rangle\} \wedge u = \tau(x)$
$\Gamma, u, \Pi, \tau \models n$ iff $\Pi = \{\langle\rangle\} \wedge u = W(n)$
$\Gamma, u, \Pi, \tau \models p$ iff $\Pi = \{\langle\rangle\} \wedge u \in V(p)$
$\Gamma, u, \Pi, \tau \models \neg\phi$ iff $\Pi = \{\langle\rangle\} \wedge \nexists\, \Pi'$ s.t. $\Gamma, u, \Pi', \tau \models \phi$
$\Gamma, u, \Pi, \tau \models \phi_1 \wedge \phi_2$ iff $\exists \Pi_1, \Pi_2$ with $\Pi_1 \cup \Pi_2 = \Pi$ s.t. $\Gamma, u, \Pi_1, \tau \models \phi_1 \wedge \Gamma, u, \Pi_2, \tau \models \phi_2$
$\Gamma, u, \Pi, \tau \models \phi_1 \vee \phi_2$ iff $\Gamma, u, \Pi, \tau \models \phi_1 \vee \Gamma, u, \Pi, \tau \models \phi_2$
$\Gamma, u, \Pi, \tau \models \langle\alpha_i\rangle\phi$ iff $\exists u' \in \mathcal{U}$ s.t. $\Gamma, u', \Pi[1:], \tau \models \phi \wedge (u, u') \in \alpha_i \wedge \forall \pi \in \Pi, \pi[0] = (u, u')$
$\Gamma, u, \Pi, \tau \models @_n\phi$ iff $\Gamma, u', \Pi, \tau \models \phi$, where $W(n) = u'$
$\Gamma, u, \Pi, \tau \models @_x\phi$ iff $\Gamma, \tau(x), \Pi, \tau \models \phi$
$\Gamma, u, \Pi, \tau \models \downarrow_x \phi$ iff $\Gamma, u, \Pi, \tau[x \mapsto u] \models \phi$

The following theorem, which we prove in the appendix, establishes that the path semantics is equivalent to the standard semantics for the hybrid logic presented in Section 2.

THEOREM 1. *For every $u \in \mathcal{U}$, $\Gamma, u, \tau \models \phi$ iff there is a set of paths Π such that $\Gamma, u, \Pi, \tau \models \phi$.*

4. RESTRICTING ACCESS IN OSNS

As stated in Section 1, adding a friend into a user's blacklist is a very useful way in OSNs for restricting the friend to access some resources of the user. Blacklists can be treated orthogonal to access control policies. In this section, we give a straightforward model of blacklists in OSNs and formally study their usage in relationship-based access control.

4.1 Blacklist in OSNs

We use a relationship type, called b, to model blacklists. If $(u, u') \in b$, then u' is on u's blacklist. For example, in Figure 2, users C and A are friends, but C is on A's blacklist.

Suppose that u_{own} has an access control policy without considering blacklist, we call this policy *non-restricted*. If he wants to restrict the policy by systematically adding the blacklist relationship to it, we say that u_{own} *blacklist-restricts* the access control policy, and the policy is a *restricted policy*.

In the examples used to motivate and illustrate our approach, we assume that the only relationships in place are *friend* (f) and *blacklist* (b). However, all our formal definitions are phrased in such a way that they apply equally when the OSN supports more relationships than these two.

4.2 Three Dimensions

Having defined the blacklist relationship in our social network model, next we focus on how to blacklist-restrict access control policies. The basic requirement is that u_{req} should never be on u_{own}'s blacklist. Beyond this requirement, there exist other decisions to make when blacklist-restricting access control policies. For instance, suppose that Alice and David share two friends Bob and Charlie, and David is on Bob's blacklist. If Alice wants to share her photo with her friends of friends and meantime forbids the access of users on her friends' blacklists, then David on one hand cannot view the photo due to his relationship with Bob, while on the other hand David can still access the photo via Charlie as he is not on Charlie's blacklist. This example shows that it is necessary to identify and precisely define how blacklists are used to restrict access in OSNs. Thanks to the path semantics of the hybrid logic in Section 3, we can classify blacklist-restrictions into three dimensions by considering the following questions: (1) whose blacklist should be used, (2) where blacklists should be applied, and (3) how many paths need to be considered. Each dimension leads to a binary decision and is defined with the reference to the paths witnessing the truth of the access control logic formula.

Whose blacklists should be used? It is clear that the blacklist of u_{own} should always be considered for blacklist-restricting policies, i.e., the user following u_{own} on a path from u_{own} to u_{req} cannot be on u_{own}'s blacklist. Besides, other users' blacklists can be considered as well. In the social graph depicted in Figure 2, suppose that user A wants to share his photo with his friends of friends. If A only considers his blacklist, then N cannot access the photo as J is on A's blacklist. If A considers the blacklists of everyone on the path, then K's access is also denied as he is on F's blacklist.

If u_{own} wants the blacklists of everyone on the path to be considered for blacklist-restricting an access control policy, u_{own} should *globally* blacklist-restrict the policy (GL). If on the other hand u_{own} only wants his own blacklist to be considered, he should *locally* blacklist-restrict the access control policy (LO). We name this restriction dimension *globality*.

Where blacklists should be applied? It is natural to require that u_{req} should never be on u_{own}'s blacklist. Besides, u_{own} may want no one on a path from him to u_{req} to be on his blacklist, i.e., he may want to consider his blacklist on the whole path. In Figure 2, suppose that A defines a 3-depth

[3]Normally the paths have the property that the end node of an edge e_i is the start node of the next edge e_{i+1} in the path. But the hybrid logic is very expressive, and for some special formulas, which in practice would hardly be used as access control policies, the satisfaction of the formula can be witnessed by a disconnected path, i.e., a path where some edge does not start where the previous edge ended.

policy. If A does not consider his blacklist on the whole path, N can access the resource due to the path (A, E, J, N). However, if A considers his blacklist on the whole path, then N's access is denied as J is on A's blacklist.

If u_{own} wants no one on a path in the set of paths witnessing the access control policy to be on his blacklist, he should perform a *general* blacklist-restriction to the policy (GE). If on the other hand u_{own} only wants u_{req} not to be on his blacklist, he should perform a *limited* blacklist-restriction to the policy (LI). We name this restriction dimension *generality*.

How many paths need to be considered? Having fixed the decisions for the previous two dimensions, u_{own} has determined which set of paths are *free of blacklist problems*. There can still be several paths from u_{own} to u_{req}, some of which are free of blacklist problems while others are not. In Figure 2, there are two 3-depth paths from A to L ((A, C, H, L) and (A, B, G, L)). Under the 3-depth policy, if A requires only one path that is free of blacklist problems, L can access the resource because of (A, B, G, L); if A requires all the paths from him to u_{req} to be free of blacklist problems, L's access is denied as (A, C, H, L) does not satisfy the local restriction.

If u_{own} just wants there to be some set of paths free of blacklist problems witnessing the access control policy, he should *weakly* blacklist-restrict the access control policy (W). If on the other hand he wants that every set of paths witnessing the policy should be free of blacklist problems, he should *strongly* blacklist-restrict the access control policy (S). We name this restriction dimension *strength*.

We now formally define the three dimensions in terms of the path semantics (Section 3). For every triple (X, Y, Z) with $X \in \{\text{LO}, \text{GL}\}$, $Y \in \{\text{LI}, \text{GE}\}$ and $Z \in \{\text{W}, \text{S}\}$ and every access control policy ϕ, we define the intended semantics of the *blacklist-restricted access control policy* $\phi_{(X,Y,Z)}$ by defining the conditions under which access is granted to u_{req} according to this blacklist-restricted access control policy. For defining these conditions, we first need to define the predicate $\mathit{Valid}_{(X,Y)}(\Pi)$, whose intended semantics is that the set Π of paths is free of blacklist problems according to the choice (X, Y) of values for the first two dimensions.

Definition 2. Let $\Gamma = (\mathcal{G}, W, V)$ be a model and τ be a valuation. For $X \in \{\text{LO}, \text{GL}\}$, $Y \in \{\text{LI}, \text{GE}\}$ and a set Π of paths, we define $\mathit{Valid}_{(X,Y)}(\Gamma, \Pi, \tau)$ to hold iff the following four properties are satisfied:

- If $X = \text{LO}$, then for every $u \in \mathcal{U}$ such that $(\tau(\mathsf{own}), u)$ is an element of some $\pi \in \Pi$, $(\tau(\mathsf{own}), u) \notin b$.
- If $X = \text{GL}$, then for all $u, u' \in \mathcal{U}$ such that (u, u') is an element of some $\pi \in \Pi$, $(u, u') \notin b$.
- If $Y = \text{LI}$, then $(\tau(\mathsf{own}), \tau(\mathsf{req})) \notin b$.
- If $Y = \text{GE}$, then $(\tau(\mathsf{own}), \tau(\mathsf{req})) \notin b$, and for all $u, u' \in \mathcal{U}$ such that (u, u') is an element of some $\pi \in \Pi$, $(\tau(\mathsf{own}), u) \notin b$ and $(\tau(\mathsf{own}), u') \notin b$.

The set of formulas not involving $\langle b \rangle$ is denoted by L'.

Definition 3. $L'(\mathsf{own}, \mathsf{req})$ is defined to be $L' \cap L(\mathsf{own}, \mathsf{req})$, i.e., the set of access control policies not containing the modality $\langle b \rangle$.

The following definition formally defines the intended semantics of the restricted access control policy $\phi_{(X,Y,Z)}$:

Definition 4. Let $\Gamma = (\mathcal{G}, W, V)$ be a model, $u \in \mathcal{U}$ and τ a valuation. Suppose $X \in \{\text{LO}, \text{GL}\}$, $Y \in \{\text{LI}, \text{GE}\}$ and $Z \in \{\text{W}, \text{S}\}$, and suppose $\phi \in L'(\mathsf{own}, \mathsf{req})$.

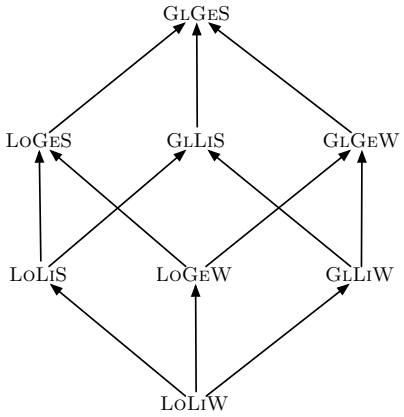


Figure 3: Black-restriction lattice

- If $Z = \text{W}$, then $\Gamma, u, \tau \vDash \phi_{(X,Y,Z)}$ iff there is a set Π of paths such that $\Gamma, u, \Pi, \tau \vDash \phi$ and $\mathit{Valid}_{(X,Y)}(\Pi)$.
- If $Z = \text{S}$, then $\Gamma, u, \tau \vDash \phi_{(X,Y,Z)}$ iff there is a set Π of paths such that $\Gamma, u, \Pi, \tau \vDash \phi$, and for every set Π of paths such that $\Gamma, u, \Pi, \tau \vDash \phi$, $\mathit{Valid}_{(X,Y)}(\Pi)$.

The eight ways of forming blacklist-restricted policies establish a lattice as shown in Figure 3. In the figure, we use, for instance, GLGES to present a restriction when the decisions in each of the three dimensions are fixed as $X = \text{GL}$, $Y = \text{GE}$, and $Z = \text{S}$. If a user's access is denied by one of the blacklist-restricted policies, then the same user's access is denied by any restricted policy above this policy in the lattice. The following theorem, which we prove in the appendix, expresses this statement formally:

THEOREM 2. *Let $\Gamma = (\mathcal{G}, W, V)$ be a model, $u \in \mathcal{U}$, τ a valuation and $\phi \in L'(\mathsf{own}, \mathsf{req})$. Let (X_1, Y_1, Z_1) and (X_2, Y_2, Z_2) be two triples with $X_1, X_2 \in \{\text{LO}, \text{GL}\}$, $Y_1, Y_2 \in \{\text{LI}, \text{GE}\}$ and $Z_1, Z_2 \in \{\text{W}, \text{S}\}$. If $(X_1, Y_1, Z_1) \leq (X_2, Y_2, Z_2)$ in the blacklist-restriction lattice, then we have that $\Gamma, u, \tau \vDash \phi_{(X_2,Y_2,Z_2)}$ implies $\Gamma, u, \tau \vDash \phi_{(X_1,Y_1,Z_1)}$.*

To illustrate the eight different blacklist-restrictions, we use the social graph in Figure 2 to present an example. We assume that the owner is A and the policy is a 3-depth policy. Under the non-restricted policy, five users including L, H, M, N and O can access the resource. Under different restrictions, different users' access are denied (see Table 1). In the following, for each restriction we explain why some users are granted and others are denied access. The complete information in Table 1 follows from these explanations by Theorem 2.

Restriction	Denied users	Restriction	Denied users
LOLIW	H	LOLIS	H, L, M
LOGEW	H, M, N	LOGES	H, L, M, N
GLLIW	H, O	GLLIS	H, L, M, O
GLGEW	H, M, N, O	GLGES	H, L, M, N, O

Table 1: Denied users under different blacklist-restrictions

1. LOLIW. Under this blacklist-restriction, H's access is denied. The only path of length 3 from A to H is (A, I, M, H), but I is on A's blacklist, so this path violates the restriction for LO.

2. LoGeW. M's access is denied. There are two paths from A to M. On the path (A, C, H, M), C is on A's blacklist which violates the restriction for Lo; on (A, D, I, M), I is on A's blacklist which violates the restriction for Ge. N's access is also denied. The only path from A to N is (A, E, J, N). On this path, J is on A's blacklist which violates the restriction for Ge.
3. GlLiW. O's access is denied. On the only path from A to O, namely (A, F, K, O), there exists a blacklist relation, namely $(F, K) \in b$, thus this path does not satisfy the restriction for Gl. M can access the resource. There is one path from A to M, namely (A, D, I, M), that does not violate the restrictions for Gl and Li.
4. GlGeW. L can access the resource. There is one path from A to L, namely (A, B, G, L), that does not violate the restrictions for Gl and Ge.
5. LoLiS. M's access is denied. There are two paths from A to M. On the path (A, C, H, M), C is on A's blacklist which violates the restriction for Lo. Since the restriction is S, M cannot access the resource. L's access is also denied. There are two paths from A to L; the path (A, C, H, L) violates the restriction for Lo, as C is on A's blacklist. Since the restriction is S, L cannot access the resource.
6. LoGeS. O can access the resource. The only path from A to O, namely (A, F, K, O), does not violate the restrictions for Lo and Ge.
7. GlLiS. N can access the resource. The only path from A to N, namely (A, E, J, N), does not violate the restrictions for Gl and Li.
8. GlGeS. No one can access the resource.

5. SYNTACTICAL TRANSFORMATION

In Section 4, we give a semantic characterization of the three dimensions for blacklist-restricting an access control policy ϕ by defining the conditions under which $\phi_{(X,Y,Z)}$ is satisfied in a given context. In this section we define an algorithm which – given an access control policy $\phi \in L'(\mathsf{own}, \mathsf{req})$ and a choice X, Y, Z of values for the three dimensions – syntactically transforms ϕ to a policy $\phi[X, Y, Z] \in L(\mathsf{own}, \mathsf{req})$ such that $\phi[X, Y, Z]$ is satisfied in precisely the same contexts as $\phi_{(X,Y,Z)}$. The model-checking algorithm from [4] can then be applied for evaluating the blacklist-restricted access control policy $\phi[X, Y, Z]$.

5.1 The Transformation Algorithm

Before presenting the algorithm that syntactically transforms ϕ to produce $\phi[X, Y, Z]$, we need to define the notion of a strictly positive subformula:

Definition 5. A subformula ψ of ϕ is *strictly positive* iff it is not in the scope of a negation symbol in ϕ.

Next we give Algorithm 1 for transforming a formula into disjunctive form, which means pulling out all strictly positive occurrences of $\vee$ in ϕ. The algorithm takes a hybrid logic formula as input and returns a list of disjuncts.

In both Algorithm 1 and Algorithm 2 for syntactically transforming ϕ, we have **for**-loops referring to subformulas of ϕ. The only requirement on the order of the iterations of these **for**-loops is that the iteration for a subformula χ of ϕ must come after the iterations of all strict subformulas of χ. Namely, we proceed from deeper to higher subformulas.

Algorithm 1 Disjunctive Form

Input: $\phi \in L$
Output: a list $DF(\phi)$ of formulas in L
1: **for** χ a strictly positive subformula of ϕ **do**
2: **if** χ is of the form $\psi_1 \wedge (\psi_2 \vee \psi_3)$ **then**
3: replace χ in ϕ by $(\psi_1 \wedge \psi_2) \vee (\psi_1 \vee \psi_3)$
4: **else if** χ is of the form $(\psi_1 \vee \psi_2) \wedge \psi_3$ **then**
5: replace χ in ϕ by $(\psi_1 \wedge \psi_3) \vee (\psi_2 \wedge \psi_3)$
6: **else if** χ is of the form $@_n(\psi \vee \chi)$ **then**
7: replace χ in ϕ by $@_n\psi \vee @_n\chi$
8: **else if** χ is of the form $@_x(\psi \vee \chi)$ **then**
9: replace χ in ϕ by $@_x\psi \vee @_x\chi$
10: **else if** χ is of the form $\langle\alpha_i\rangle(\psi \vee \chi)$ **then**
11: replace χ in ϕ by $\langle\alpha_i\rangle\psi \vee \langle\alpha_i\rangle\chi$
12: **else if** χ is of the form $\downarrow_x(\psi \vee \chi)$ **then**
13: replace χ in ϕ by $\downarrow_x\psi \vee \downarrow_x\chi$
14: $DF(\phi) \leftarrow \{\psi \mid \psi$ is a disjunct of $\phi\}$

Algorithm 2 takes a formula ϕ as input and syntactically transforms it to $\phi[X, Y, Z] \in L(\mathsf{own}, \mathsf{req})$. The transformation is defined separately for weak blacklist-restrictions (lines 2-12) and strong blacklist-restrictions (lines 13-27). In both cases, we insert $\downarrow_{x_k}$'s and $\downarrow_{y_k}$'s into the formula (lines 3 and 15) in order to be able to refer to the nodes of the paths satisfying the formula. We then use the bound variables x_k, y_k to formulate the conditions of Definition 2 to ensure that the specified blacklist-restriction in $[X, Y, Z]$ is satisfied.

The following theorem, whose proof is sketched in the appendix, establishes the equivalence between the syntactical transformation $\phi[X, Y, Z]$ and the semantically defined satisfaction conditions for $\phi_{(X,Y,Z)}$:

Theorem 3. *Let $\Gamma = (\mathcal{G}, W, V)$ be a model, $u \in \mathcal{U}$, τ a valuation and $\phi \in L'(\mathsf{own}, \mathsf{req})$. Let $X \in \{\mathrm{Lo}, \mathrm{Gl}\}$, $Y \in \{\mathrm{Li}, \mathrm{Ge}\}$ and $Z \in \{\mathrm{W}, \mathrm{S}\}$. Then $\Gamma, u, \tau \vDash \phi[X, Y, Z]$ iff $\Gamma, u, \tau \vDash \phi_{(X,Y,Z)}$.*

We illustrate the syntactical transformation by showing its results for some typical policies and blacklist-restrictions:

$@_{\mathsf{own}}\langle f\rangle\langle f\rangle\mathsf{req}[\mathrm{Gl}, \mathrm{Li}, \mathrm{W}] =$
$@_{\mathsf{own}} \downarrow_{x_1} \langle f\rangle \downarrow_{y_1}\downarrow_{x_2} \langle f\rangle \downarrow_{y_2} (\mathsf{req} \wedge \neg@_{x_1}\langle b\rangle y_1 \wedge$
$\neg@_{x_2}\langle b\rangle y_2) \wedge \neg@_{\mathsf{own}}\langle b\rangle\mathsf{req}$

$@_{\mathsf{own}}\langle f\rangle\langle f\rangle\mathsf{req}[\mathrm{Lo}, \mathrm{Ge}, \mathrm{S}] =$
$@_{\mathsf{own}}\langle f\rangle\langle f\rangle\mathsf{req} \wedge \neg@_{\mathsf{own}} \downarrow_{x_1} \langle f\rangle \downarrow_{y_1}\downarrow_{x_2} \langle f\rangle \downarrow_{y_2} (\mathsf{req} \wedge$
$((@_{\mathsf{own}}x_1 \wedge @_{x_1}\langle b\rangle y_1) \vee (@_{\mathsf{own}}x_2 \wedge @_{x_2}\langle b\rangle y_2) \vee @_{\mathsf{own}}\langle b\rangle x_1 \vee$
$@_{\mathsf{own}}\langle b\rangle y_1 \vee @_{\mathsf{own}}\langle b\rangle x_2 \vee @_{\mathsf{own}}\langle b\rangle y_2)) \wedge \neg@_{\mathsf{own}}\langle b\rangle\mathsf{req}$

5.2 Blacklist-restriction in Practice

Allowing the users to write access control policies in a hybrid logic gives them a lot of flexibility in the specification of the policies. But in practice, if one has in mind an OSN whose users are not all computer scientists, logicians or mathematicians, one cannot expect users to be or become competent in writing formulas in hybrid logic. Instead, we envisage an OSN to provide a tool to the users that allows them to specify an access control policy in an easy-to-understand and hence user-friendly way. This tool would produce a hybrid logic formula to be used internally. Such a tool would give the user various options for considering various information in the access control policy and for

Algorithm 2 Syntactical Transformation

Input: $\phi \in L'(\mathsf{own}, \mathsf{req})$, $X \in \{\textsc{Lo}, \textsc{Gl}\}$, $Y \in \{\textsc{Li}, \textsc{Ge}\}$, $Z \in \{\mathrm{W}, \mathrm{S}\}$

let $x_1, y_1, x_2, y_2, \ldots$ be variables not occurring in ϕ
if $Z = \mathrm{W}$ **then**
  replace every strictly positive subformula of ϕ of the form $\langle\alpha_i\rangle\psi$ by $\downarrow_{x_k} \langle\alpha_i\rangle \downarrow_{y_k} \psi$.
  for χ a strictly positive subformula of ϕ of the form x, n, p or $\neg\psi$ **do**
    $K_\chi \leftarrow \{k \mid$ some subformula $\downarrow_{x_k} \psi$ of ϕ contains $\chi\}$
  if $X = \textsc{Lo}$ **then**
    replace every strictly positive subformula χ of ϕ of the form x, n, p or $\neg\psi$ by $\chi \wedge \bigwedge_{k\in K_\chi}(@_{\mathsf{own}} x_k \rightarrow \neg @_{x_k}\langle b\rangle y_k)$
  if $X = \textsc{Gl}$ **then**
    replace every strictly positive subformula χ of ϕ of the form x, n, p or $\neg\psi$ by $\chi \wedge \bigwedge_{k\in K_\chi} \neg @_{x_k}\langle b\rangle y_k$
  if $Y = \textsc{Ge}$ **then**
    replace every strictly positive subformula χ of ϕ of the form x, n, p or $\neg\psi$ by $\chi \wedge \bigwedge_{k\in K_\chi}(\neg @_{\mathsf{own}}\langle b\rangle x_k \wedge \neg @_{\mathsf{own}}\langle b\rangle y_k)$
  $\psi \leftarrow \phi \wedge \neg @_{\mathsf{own}}\langle b\rangle \mathsf{req}$
if $Z = \mathrm{S}$ **then**
  for $\phi_i \in (DF(\phi))$ **do**
    replace every strictly positive subformula of ϕ_i of the form $\langle\alpha_i\rangle\psi$ by $\downarrow_{x_k} \langle\alpha_i\rangle \downarrow_{y_k} \psi$.
    for $\chi_{i,j}$ a strictly positive subformula of ϕ_i of the form x, n, p or $\neg\psi$ **do**
      $K_{\chi_{i,j}} \leftarrow \{k |$some subformula $\downarrow_{x_k} \psi$ of ϕ contains $\chi\}$
      if $X = \textsc{Lo}$ **then**
        $\psi_{i,j} \leftarrow \bigvee_{k\in K_{\chi_{i,j}}}(@_{\mathsf{own}} x_k \wedge @_{x_k}\langle b\rangle y_k)$
      if $X = \textsc{Gl}$ **then**
        $\psi_{i,j} \leftarrow \bigvee_{k\in K_{\chi_{i,j}}} @_{x_k}\langle b\rangle y_k$
      if $Y = \textsc{Ge}$ **then**
        $\psi_{i,j} \leftarrow \psi_{i,j} \vee \bigvee_{k\in K_{\chi_{i,j}}}(@_{\mathsf{own}}\langle b\rangle x_k \vee @_{\mathsf{own}}\langle b\rangle y_k)$
    $\phi_{i,j} \leftarrow$ result of replacing $\chi_{i,j}$ in ϕ_i by $\chi_{i,j} \wedge \psi_{i,j}$
    $\overline{\phi}_i \leftarrow \bigwedge_j \neg\phi_{i,j}$
  $\overline{\phi} \leftarrow \phi \wedge \bigwedge_i \overline{\phi}_i$
  $\psi \leftarrow \overline{\phi} \wedge \neg @_{\mathsf{own}}\langle b\rangle \mathsf{req}$
$\phi[X, Y, Z] \leftarrow \psi$

making the policy more stringent or more lax. One of the decisions that a user has to make is whether and how to use the information from his and other users' blacklists. The three dimensions discussed in the previous section constitute three binary choices of how to use blacklist information in the policy. We believe that these three binary choices are simple enough to make them comprehensible to non-expert users.

As we have seen in first part of this section, for every access control policy ϕ not involving the modality $\langle b\rangle$ and any choice of X, Y, Z for the three dimensions, there is an access control policy $\phi[X, Y, Z] \in L(\mathsf{own}, \mathsf{req})$ such that $\phi[X, Y, Z]$ is satisfied in precisely the same contexts as $\phi_{(X,Y,Z)}$. In other words, the three dimensions for blacklist-restriction do not allow us to express any policy that is not already expressible in the hybrid logic with the help of the modality $\langle b\rangle$. But even if we assume the users to have some competence in writing hybrid logic formulas, it would be cumbersome for the users to write $\phi[X, Y, Z]$ themselves, for often $\phi[X, Y, Z]$ is much more complex than ϕ. Possibly in combination with some tool for producing the basic formula ϕ, our approach can be used for allowing users to flexibly use the information from the blacklists for restricting their access control policies without the need to write complex hybrid logic formulas. This makes our approach a *user-friendly* framework for restricting access in social networks.

Algorithm 3 Path Policy Evaluation

Input: u_{own}, u_{req}, $\mathcal{G}$, $\phi \in L'(\mathsf{own}, \mathsf{req})$, $X \in \{\textsc{Lo}, \textsc{Gl}\}$, $Y \in \{\textsc{Li}, \textsc{Ge}\}$, $Z \in \{\mathrm{W}, \mathrm{S}\}$
Output: access permission

if $(u_{\mathsf{own}}, u_{\mathsf{req}}) \in b$ **then**
  access denied
else
  if $Z = \mathrm{W}$ **then**
    for each path policy ϕ' of ϕ **do**
      extract rp and n from ϕ'
      $satisfied_{\phi'} \leftarrow \mathsf{Weak}(u_{\mathsf{own}}, u_{\mathsf{req}}, \mathcal{G}, rp, n, X, Y)$
      if $satisfied_{\phi'} = 1$ **then**
        access granted, **return**
    if access permission is not set **then**
      access denied
  else if $Z = \mathrm{S}$ **then**
    for each path policy ϕ' of ϕ **do**
      extract rp and n from ϕ'
      $(nopath_{\phi'}, satisfied_{\phi'}) \leftarrow \mathsf{Strong}(u_{\mathsf{own}}, u_{\mathsf{req}}, \mathcal{G}, rp, n, X, Y)$
      if $satisfied_{\phi'} = 0$ **then**
        access denied, **return**
    if $\bigwedge_{(\phi' \text{ of } \phi)} nopath_{\phi'} = 1$ **then**
      access denied
    else
      access granted

6. PATH EVALUATION ALGORITHMS

In practice, especially in the most popular OSNs such as Facebook, a user normally focuses on the length of the path between him and the potential requesters when defining his access control policies. In Facebook one could define a policy to allow his friends or friends of friends to view his profile. In the hybrid logic, the policy can be represented as $@_{\mathsf{own}}\langle f\rangle\mathsf{req} \vee @_{\mathsf{own}}\langle f\rangle\langle f\rangle\mathsf{req}$.

To evaluate this formula under a blacklist-restriction, we can follow the procedure as described in Section 5 to transform the policy into a blacklist-restricted policy. Then we apply the local model-checking algorithm of Bruns et al. [4] to evaluate the resulting policy on a social network model. However, as we have seen with the two examples in Section 5, after the transformation the size of the new formula is usually getting larger, which in turn will make the evaluation using model-checking inefficient: The model checking algorithm needs to go through the structure of the formula (see details in [4]).

In fact, to evaluate a policy that only focuses on the path length from u_{own} to u_{req}, we can first decompose it into several sub-policies, e.g., $@_{\mathsf{own}}\langle f\rangle\mathsf{req}$ and $@_{\mathsf{own}}\langle f\rangle\langle f\rangle\mathsf{req}$ for the above policy, and evaluate each sub-policy by finding the qualified path(s) from u_{own} to u_{req}. During the path-finding process, we can perform optimizations such as filtering out the users who are on u_{own}'s blacklist on-the-fly. In the end,

Algorithm 4 Weak

Input: u_{own}, u_{req}, $\mathcal{G}$, rp, n, $X \in \{\text{Lo}, \text{Gl}\}$, $Y \in \{\text{Li}, \text{Ge}\}$
Output: *satisfied*

1: *ulist* $\leftarrow \{u \mid (u_{\mathsf{own}}, u) \in rp(1) \wedge (u_{\mathsf{own}}, u) \notin b\}$
2: **if** $[X, Y] =$ LoLi **then**
3: **for** $i = 2 : n-1$ **do**
4: **for** $u \in$ *ulist* **do**
5: add $\{u' | (u, u') \in rp(i)\}$ into *ulist*
6: delete u from *ulist*
7: **for** $u \in$ *ulist* **do**
8: **if** $(u, u_{\mathsf{req}}) \in rp(n)$ **then**
9: *satisfied* $\leftarrow 1$, **break**
10: **else if** $[X, Y] =$ LoGe **then**
11: **for** $i = 2 : n-1$ **do**
12: **for** $u \in$ *ulist* **do**
13: add $\{u' | (u, u') \in rp(i) \wedge (u_{\mathsf{own}}, u') \notin b\}$ into *ulist*
14: delete u from *ulist*
15: **for** $u \in$ *ulist* **do**
16: **if** $(u, u_{\mathsf{req}}) \in rp(n) \wedge (u_{\mathsf{own}}, u_{\mathsf{req}}) \notin b$ **then**
17: *satisfied* $\leftarrow 1$, **break**
18: **else if** $[X, Y] =$ GlLi **then**
19: **for** $i = 2 : n-1$ **do**
20: **for** $u \in$ *ulist* **do**
21: add $\{u' | (u, u') \in rp(i) \wedge (u, u') \notin b\}$ into *ulist*
22: delete u from *ulist*
23: **for** $u \in$ *ulist* **do**
24: **if** $(u, u_{\mathsf{req}}) \in rp(n) \wedge (u, u_{\mathsf{req}}) \notin b$ **then**
25: *satisfied* $\leftarrow 1$, **break**
26: **else if** $[X, Y] =$ GlGe **then**
27: **for** $i = 2 : n-1$ **do**
28: **for** $u \in$ *ulist* **do**
29: add $\{u' | (u, u') \in rp(i) \wedge (u, u') \notin b \wedge (u_{\mathsf{own}}, u') \notin b\}$ into *ulist*
30: delete u from *ulist*
31: **for** $u \in$ *ulist* **do**
32: **if** $(u, u_{\mathsf{req}}) \in rp(n) \wedge (u, u_{\mathsf{req}}) \notin b \wedge (u_{\mathsf{own}}, u_{\mathsf{req}}) \notin b$ **then**
33: *satisfied* $\leftarrow 1$, **break**
34: **if** *satisfied* is not set **then**
35: *satisfied* $\leftarrow 0$

Algorithm 5 Strong

Input: u_{own}, u_{req}, $\mathcal{G}$, rp, n, $X \in \{\text{Lo}, \text{Gl}\}$, $Y \in \{\text{Li}, \text{Ge}\}$
Output: *nopath*, *satisfied*

1: *path* $\leftarrow \mathsf{BFS}(u_{\mathsf{own}}, u_{\mathsf{req}}, \mathcal{G}, rp, n)$
2: **if** *path* is empty **then**
3: *nopath* $\leftarrow 1$, *satisfied* $\leftarrow 1$
4: **else**
5: *nopath* $\leftarrow 0$
6: **if** $[X, Y] =$ LoLi **then**
7: **for** $p \in$ *path* **do**
8: **if** $(u_{\mathsf{own}},$ the first user on $p) \in b$ **then**
9: *satisfied* $\leftarrow 0$, **break**
10: **else if** $[X, Y] =$ LoGe **then**
11: **for** $p \in$ *path* **do**
12: **if** $\exists$ a user u on p s.t. $(u_{\mathsf{own}}, u) \in b$ **then**
13: *satisfied* $\leftarrow 0$, **break**
14: **else if** $[X, Y] =$ GlLi **then**
15: **for** $p \in$ *path* **do**
16: **if** $\exists (u, u')$ is part of p s.t. $(u, u') \in b$ **then**
17: *satisfied* $\leftarrow 0$, **break**
18: **else if** $[X, Y] =$ GlGe **then**
19: **for** $p \in$ *path* **do**
20: **if** $\exists u$ on p s.t. $(u_{\mathsf{own}}, u) \in b \vee \exists (u, u')$ is part of p s.t. $(u, u') \in b$ **then**
21: *satisfied* $\leftarrow 0$, **break**
22: **if** *satisfied* is not set **then**
23: *satisfied* $\leftarrow 1$

access permission is made by the result of the boolean function connecting the results of each sub-policy. In this way, for policies of such simple form, we can avoid syntactical transformation as well as model-checking, and design more efficient algorithms for policy evaluation.

The policies we consider here can be written as the disjunctions of several *path policies*, and each path policy has the form of $@_{\mathsf{own}}\langle\alpha_1\rangle \ldots \langle\alpha_n\rangle\mathsf{req}$, representing a certain depth path from u_{own} to u_{req}. Among the three dimensions, both *globality* and *generality* concentrate on how blacklists are used on a single path while *strength* takes into account all the paths from u_{own} to u_{req}. When the policy's blacklist-restriction is weak, u_{req} can access u_{own}'s resource as long as there exists a path that satisfies the restrictions from the other two dimensions. Therefore, during the process of finding paths from u_{own} to u_{req}, we can directly skip the unqualified edges. On the other hand, when the restriction is strong, we need to make sure that all the possible paths from u_{own} to u_{req} are free of blacklist problems. Since the processes for evaluating weak and strong restrictions are different, we treat them separately.

Our evaluation algorithm is listed in Algorithm 3. Its input consists of u_{own}, u_{req}, a policy ϕ and a blacklist-restriction X, Y, Z. Due to the restriction of the *generality* dimension, we first check whether u_{req} is on u_{own}'s blacklist. If he is, then we directly deny his access (lines 1-2). Otherwise, we check path policies one by one. Depending on the strength restriction of each path policy, we use the corresponding algorithm (lines 4-17). Each path policy represents a relation path denoted by rp. Here, $rp = (\alpha_1, \ldots \alpha_n)$ is tuple with each item as the corresponding relationship type specified in the path policy and it is indexed by $rp(i)$. Moreover, n is the length of the path (lines 6 and 12). Under the weak restriction, once a path policy's evaluation result is positive ($satisfied_{\phi'} = 1$), u_{req}'s access is granted (lines 8-9). Under the strong restriction, if there exists no path (specified in all the path policies) from u_{own} to u_{req}, u_{req}'s access is denied (lines 18-19). Otherwise, all the existing paths from u_{own} to u_{req} have to satisfy the restrictions from the other two dimensions. If one path policy is not satisfied, then the access is denied and the algorithm is finished (lines 16-17).

Algorithm 4 is used for evaluating path policies under weak restrictions. Here, we perform breadth first search (BFS) to find paths from u_{own} to u_{req} in the social graph $\mathcal{G}$. We first add u_{own}'s $rp(1)$ relations who are not on his blacklist into a list *ulist*, thus the local restriction is implemented. Then, depending on the chosen restriction, different processes are conducted. For example, when the restriction is LoGeW, for each user, to traverse his friends, we only consider the ones that are not on u_{own}'s blacklist (line 13). Note that in the last step, once there is a qualified path

from a user in *ulist* to u_{req}, the access is directly granted (*satisfied*$\leftarrow$1) (e.g., lines 15-17).

Algorithm 5 presents the process for evaluating the policies under strong restrictions. In the beginning, we exploit BFS to find all the paths from u_{own} to u_{req}. If there is no path from u_{own} to u_{req}, then *nopath* is set to 1 (line 3). Otherwise, we begin to evaluate the paths. Under strong restrictions, once we find an unqualified path from u_{own} to u_{req}, we can directly deny u_{req}'s access without considering other paths anymore (*satisfied* $\leftarrow$ 0). For example, under restriction LoLiS, as long as there exists one path whose first user is on u_{own}'s blacklist, the access is denied (lines 8-9).

7. EVALUATION

As introduced in Section 6, our path evaluation algorithms only consider access control policies that are composed by one or several path policies and each path policy represents a relation path from u_{own} to u_{req}. For empirical evaluation, we focus on the 2-depth policy and the 3-depth policy.

7.1 Algorithm Efficiency

Experiment setup. To evaluate the performance of our proposed algorithms in Section 6, we check the time difference between evaluating restricted and non-restricted policies. The metric we adopt is defined as $time\_ratio[X, Y, Z] = t[X, Y, Z]/t$, where t is the time for checking a non-restricted policy and $t[X, Y, Z]$ is the time for checking the corresponding restricted policy. Here, to enforce a non-restricted policy, we perform BFS to find whether there exists a path from u_{own} to u_{req} satisfying the policy. Since major OSN companies such as Facebook do not disclose their algorithms for enforcing access control policies, we simply choose BFS for the purpose to evaluate the performance of our algorithms. Other algorithms for path-finding can be used as well.

The dataset we use to conduct our experiments is collected by McAuley and Leskovec [17], it is a Facebook dataset that contains 4,039 users and 88,234 edges. For each user, we randomly sample five different ratios, i.e., 1%, 5%, 10%, 20% and 30% of his friends to be on his blacklist. The ratio is called the *blacklist ratio*. The algorithms are implemented on a machine with Intel core i7 processor and 8GB RAM.

Experimental results. For each blacklist-restriction, we plot the metric *time_ratio* as a function of blacklist ratio in Figure 4. The performance of algorithms is quite different for weak and strong restrictions.

Weak restrictions. As shown in Figures 4a, 4b, 4c and 4d, with the increase of blacklist ratio, checking path policies under weak restrictions is getting faster. This is because during the path-finding process, Algorithm 4 filters out all the unqualified edges which saves a lot of operations. On the other hand, for non-restricted policies, the algorithm cannot skip any edges until it finds a path. Due to the same reason, evaluating weak restrictions is faster than evaluating strong ones. We also notice that, in Figures 4b, 4c, 4d, the curves for 3-depth policies (blue) are far below the curves for 2-depth ones (red). The reason is that longer paths our algorithm traverses, more edges it filters out, thus more operations are saved compared to running non-restricted policies. On the other hand, the difference between the two curves in Figure 4a is small since running LoLiW only filters out the users that are on u_{own}'s blacklist in the first step.

Strong restrictions. As depicted in Figures 4e, 4f, 4g and 4h, time for running 3-depth policies with strong restrictions is almost twice as much as running non-restricted policies. This indicates that the most time-consuming operations are for finding paths. On the other hand, checking 2-depth strong policies only requires around 30% overhead.

7.2 Power of Blacklist-restrictions

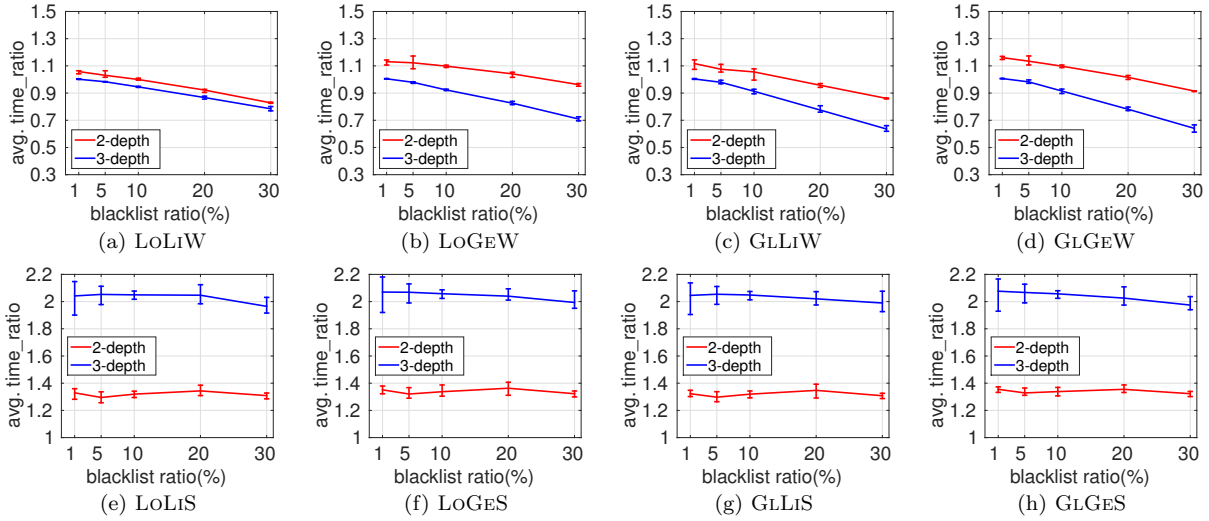
It is interesting to learn what is the impact of different restrictions on access control. We focus on two questions.

Which restrictions are relatively powerful? The "power" of a blacklist-restriction is quantified by the number of users denied by it. We first define a metric, *access_ratio*, representing the fraction of the number of qualified requesters under an owner's restricted policy and the number of qualified requesters under the same non-restricted policy. When a user's *access_ratio* under a blacklist-restriction is high, it means that he *cannot* forbid many users with the restriction.

As we can see from Figure 5, the power of all the eight blacklist-restrictions is consistent with the lattice presented in Figure 3. GlGeS which is the supremum in the lattice is the most powerful blacklist-restriction. When the blacklist ratio is 20%, the average *access_ratio* is only 20% (40%) for the 3-depth (2-depth) case (see Figure 5h). On the other hand, LoLiW is the least powerful one. When the blacklist ratio is 20%, the average *access_ratio* is around 85% for the 3-depth case (see Figure 5a). For each edge of the lattice in Figure 3, the restriction of the source node always denies less users than the one of the target node, e.g., LoGeW denies less users than LoGeS (Figure 5b vs. Figure 5f).

We notice that among all the three dimensions, shifting the *strength* dimension from weak to strong denies many more users' access than shifting the other two dimensions. For example, the difference between the curves in Figure 5f (LoGeW) and Figure 5b (LoGeS) is much bigger than the difference between Figure 5b (LoGeW) and Figure 5d (GlGeW). This is because the strong restriction requires all the paths from u_{own} to u_{req} to be free of blacklist problems, while the weak restriction only needs one qualified path. On the other hand, shifting the *globality* dimension from local to global denies more users than shifting the *generality* from limited to general. For example, by shifting the blacklist-restriction from GlLiW to GlGeW, the *access_ratio* barely changes (see Figure 5c and Figure 5d), while the difference between LoLiS and GlLiS is more notable (see Figure 5e and Figure 5g). The reason is that the global restriction considers the blacklist of everyone on the path from u_{own} to u_{req} while the general restriction only focuses on u_{own}'s blacklist.

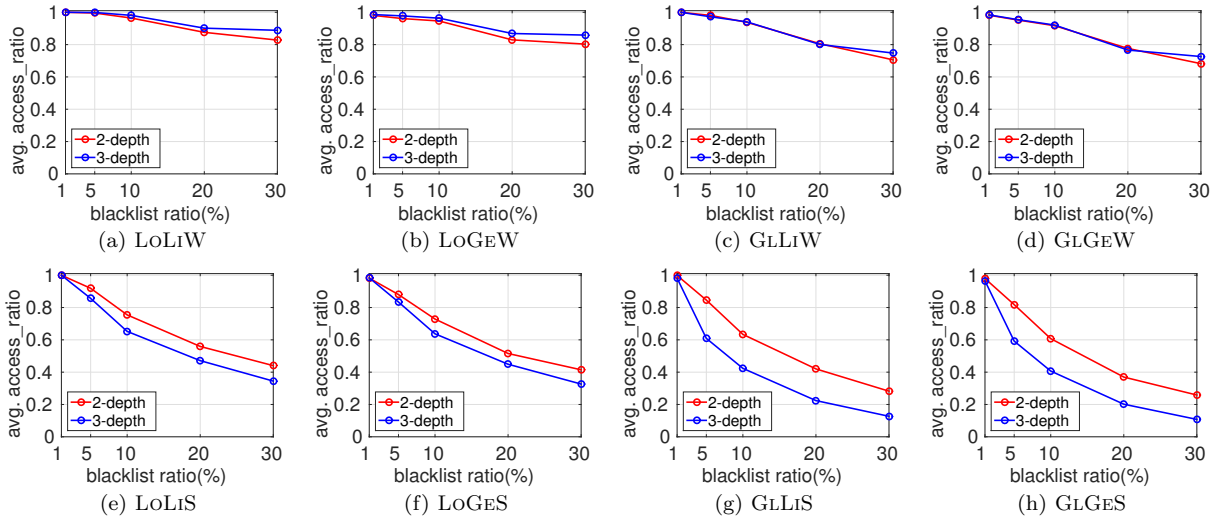
Which users are relatively easily to be forbidden? To precisely answer this question, we study the social strength between the owner and the qualified requesters under different blacklist-restrictions. The social strength between two users is quantified by three metrics including embeddedness, Jaccard index and Adamic-Adar score [1]. If two users' embeddedness (as well as Jaccard index and Adamic-Adar score) is high, then they are considered to have a strong relationship. We compute the average value of the three metrics between the qualified requesters and the corresponding owners under different blacklist-restrictions. As shown in Figure 6, the three metrics give us similar results. Qualified requesters under weak restrictions are more socially close to the corresponding owners than the qualified requesters under strong ones. This is because higher social strength implies more



(a) LoLiW (b) LoGeW (c) GlLiW (d) GlGeW

(e) LoLiS (f) LoGeS (g) GlLiS (h) GlGeS

Figure 4: Average *time_ratio* under eight blacklist-restrictions



(a) LoLiW (b) LoGeW (c) GlLiW (d) GlGeW

(e) LoLiS (f) LoGeS (g) GlLiS (h) GlGeS

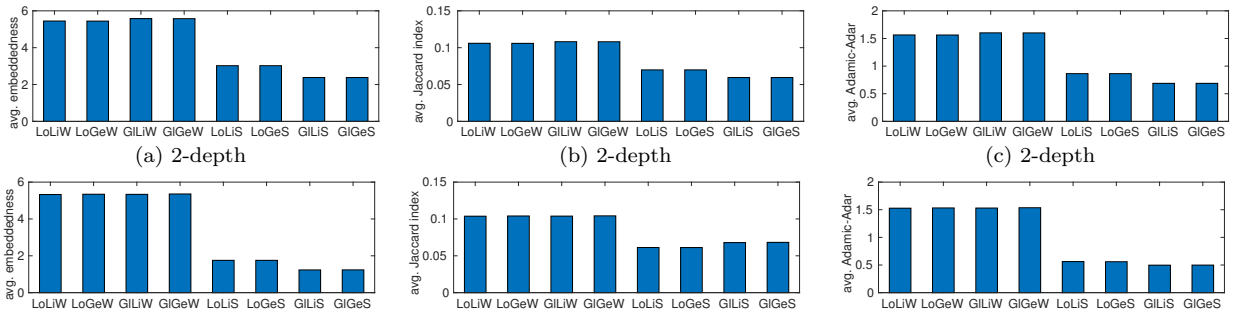
Figure 5: Average *access_ratio* under eight blacklist-restrictions

paths. Therefore, there is a better chance for the requester to be qualified under weak restrictions. However, to access the resource under a strong blacklist-restriction, the requester is better *not* to be socially close with the owner, which seems counter-intuitive. This is because the strong restriction considers all the paths from u_{own} to u_{req}.

8. RELATED WORK

Blacklists have been used in a wide range of applications, such as spam detection [7, 19] and sybil defense [22, 11]. In this work, we focus on the use of blacklists in relationship-based access control.

Relationship-based access control is first proposed in [14], it states that the data owner can control the access to his data based on the relationship between him and the requester. Following this work, several papers have focused on modeling relationship-based access control systems. In [6], Carminati et al. interpret the access decision in terms of three conditions including relationship, depth and trust level between the owner and the requester. In [13], the authors model the relationship-based access control into a two-stage process where the requester needs to first be able to reach the owner and then applies for access. Besides proposing different models, defining and specifying access control policies have also been studied in the literature. In [5], the authors exploit the use of semantic web to define policies. Moreover, they propose three system-level policies including authorization, admin and filtering policies. Fong et al. [13] propose several topology-based policies such as k-common friends and clique. Later, the authors of [12, 4] exploit hybrid logics to specify these fine-grained policies. Their logic is quite expressive and has been used in several other systems [20, 21, 18], and we adopt the same logic to specify restricted access. In [8], Cheng et al. consider not only user-to-user, but also user-to-resource and resource-to-resource relationships in OSNs, which enables them to express new types of policies such as users who are tagged in the same photo with the owner can view his profile can be expressed. Crampton and Sellwood [10] apply the ideas of relationship-based access control on general computing systems. They propose



(a) 2-depth (b) 2-depth (c) 2-depth

(d) 3-depth (e) 3-depth (f) 3-depth

Figure 6: Embeddedness, Jaccard index and Adamic-Adar score between users and the owner (blacklist ratio = 10%)

path conditions to specify policies and principal matching for evaluation. However, to the best of our knowledge, exploring blacklists for restricting access has not been discussed in relationship-based access control. Moreover, it is the first time to model and formally define different blacklist-restrictions in a hybrid logic.

Delegation means that one active entity in a system delegates its authority to another entity in the systems to carry out some functions; it has been extensively studied in access control (e.g., see [23, 16, 2, 9]). Fong [12] explicitly points out that relationship-based access control supports delegation – the use of other users' social relations (and blacklists) to regulate access control in OSNs can be naturally considered as a delegation process. Revocation is an important issue that has been studied with delegation [23], which has been formally categorized and defined in [3, 15]. When a user blacklist-restricts a policy, it can be treated as revoking other users' privileges that they are delegated under the corresponding non-restricted policy. For example, under the restriction GlLiW, a user can only delegate privileges to his friends that are not on his blacklist. Different from revocation which takes away all the delegated users' privileges, blacklist-restrictions can be considered a "partial" revocation since a user can still delegate the privilege to others if the blacklist-restrictions are not violated. The formal relation between blacklist-restriction and revocation deserves further investigations, and we leave it for our future work.

9. CONCLUSION

In this paper, we have focused on blacklists, which already exist in popular OSNs such as Facebook, for the purpose of restricting access. We treated blacklists as a special relationship among OSN users. This allows us to build our work naturally on an exiting social network model and a hybrid logic for specifying relationship-based access control policies. We have identified three different dimensions of applying blacklists. Each dimension provides a binary choice, resulting into eight types of blacklist-restrictions. The meaning of the choices are intuitive for the users to understand. We formally defined the blacklist-restrictions, using a new path semantics for the hybrid logic. To release users from the task of precisely writing their policies with blacklist-restrictions and in order to make our approach user-friendly, we also provided a procedure to syntactically rewrite a non-restricted policy into a policy under a user specified blacklist-restriction. To enforce policies which require the witness of a relation path from the owner to the requester, we designed efficient algorithms for blacklist-restrictions and evaluated their performance on a Facebook dataset. In addition, we have made a few interesting observations on the impact of the blacklist-restrictions for access control in OSNs.

10. ACKNOWLEDGEMENTS

The work of Marcos Cramer was supported by the FNR INTER project *Specification logics and Inference tools for verification and Enforcement of Policies*.

11. REFERENCES

[1] L. Adamic and E. Adar. Friends and neighbors on the web. *Social Networks*, 25(3):211–230, 2003.

[2] M. Alam, X. Zhang, K. Khan, and G. Ali. xDAuth: a scalable and lightweight framework for cross domain access control and delegation. In *Proc. SACMAT*, pages 31–40. ACM, 2011.

[3] E. Barka and R. S. Sandhu. Framework for role-based delegation models. In *Proc. ACSAC*, pages 168–176. IEEE CS, 2000.

[4] G. Bruns, P. W. L. Fong, I. Siahaan, and M. Huth. Relationship-based access control: its expression and enforcement through hybrid logic. In *Proc. CODASPY*, pages 117–124. ACM, 2012.

[5] B. Carminati, E. Ferrari, R. Heatherly, M. Kantarcioglu, and B. Thuraisingham. A semantic web based framework for social network access control. In *Proc. SACMAT*, pages 177–186. ACM, 2009.

[6] B. Carminati, E. Ferrari, and A. Perego. Rule-based access control for social networks. In *Proc. IFIP WG 2.12 and 2.14 Semantic Web Workshop (OTM)*, volume 4278 of *LNCS*, pages 1734–1744. Springer, 2006.

[7] C. Castillo, D. Donato, A. Gionis, V. Murdock, and F. Silvestri. Know your neighbors: web spam detection using the web topology. In *Proc. SIGIR*, pages 423–430. ACM, 2007.

[8] Y. Cheng, J. Park, and R. S. Sandhu. Relationship-based access control for online social networks: beyond user-to-user relationships. In *Proc. PASSAT*, pages 646–655. IEEE CS, 2012.

[9] J. Crampton and C. Morisset. An auto-delegation mechanism for access control systems. In *Proc. STM*, volume 6710 of *LNCS*, pages 1–16. Springer, 2011.

[10] J. Crampton and J. Sellwood. Path conditions and principal matching: a new approach to access control. In *Proc. SACMAT*, pages 187–198. ACM, 2014.

[11] P. W. L. Fong. Preventing sybil attacks by privilege attenuation: a design principle for social network systems. In *Proc. S&P*, pages 263–278. IEEE CS, 2011.
[12] P. W. L. Fong. Relationship-based access control: protection model and policy language. In *Proc. CODASPY*, pages 191–202. ACM, 2011.
[13] P. W. L. Fong, M. M. Anwar, and Z. Zhao. A privacy preservation model for Facebook-style social network systems. In *Proc. ESORICS*, volume 5789 of *LNCS*, pages 303–320. Springer, 2009.
[14] C. E. Gates. Access control requirements for Web 2.0 security and privacy. In *Proc. IEEE Workshop on Web 2.0 Security and Privacy (W2SP)*, 2007.
[15] A. Hagstrom, S. Jajodia, F. Parisi-Presicce, and D. Wijesekera. Revocations-a classification. In *Proc. CSFW*. IEEE CS, 2001.
[16] J. Joshi and E. Bertino. Fine-grained role-based delegation in presence of the hybrid role hierarchy. In *Proc. SACMAT*, pages 81–80. ACM, 2006.
[17] J. J. McAuley and J. Leskovec. Learning to discover social circles in ego networks. In *Proc. NIPS*, pages 548–556. NIPS, 2012.
[18] J. Pang and Y. Zhang. A new access control scheme for Facebook-style social networks. In *Proc. ARES*, pages 1–10. IEEE CS, 2014.
[19] A. Ramachandran, N. Feamster, and S. Vempala. Filtering spam with behavioral blacklisting. In *Proc. CCS*, pages 342–351. ACM, 2007.
[20] E. Tarameshloo and P. W. L. Fong. Access control models for geo-social computing systems. In *Proc. SACMAT*, pages 115–126. ACM, 2014.
[21] E. Tarameshloo, P. W. L. Fong, and P. Mohassel. On protection in federated social computing systems. In *Proc. CODASPY*, pages 75–86. ACM, 2014.
[22] H. Yu, P. B. Gibbons, M. Kaminsky, and F. Xiao. Sybillimit: A near-optimal social network defense against sybil attacks. In *Proc. S&P*, pages 3–17. IEEE CS, 2008.
[23] L. Zhang, G.-J. Ahn, and B.-T. Chu. A rule-based framework for role-based delegation and revocation. *ACM TIST*, 6(3):404–441, 2003.

APPENDIX

A. PROOF OF THEOREM 1

We proof the theorem by induction over the length of ϕ.

- $\phi = x$:
  Left-to-right: Suppose $\Gamma, u, \tau \vDash x$, i.e. $u = \tau(x)$. Set $\Pi := \{\langle\rangle\}$. Then $\Gamma, u, \Pi, \tau \vDash x$.
  Right-to-left: Trivial.
- $\phi = n$: Similar to the case $\phi = x$
- $\phi = p$: Similar to the case $\phi = x$
- $\phi = \neg\psi$:
  Left-to-right: Suppose $\Gamma, u, \tau \vDash \neg\psi$, i.e. $\Gamma, u, \tau \nvDash \psi$. By the inductive hypothesis, there is no set of paths Π' s.t. $\Gamma, u, \Pi', \tau \vDash \phi$. Set $\Pi := \{\langle\rangle\}$. It now follows that $\Gamma, u, \Pi, \tau \vDash \neg\psi$.
  Right-to-left: Suppose $\Gamma, u, \Pi, \tau \vDash \neg\psi$, i.e. $\Pi = \{\langle\rangle\}$ and there is no set of paths Π' s.t. $\Gamma, u, \Pi', \tau \vDash \phi$. By the inductive hypothesis, $\Gamma, u, \tau \vDash \neg\psi$.
- $\phi = \psi_1 \wedge \psi_2$:
  Left-to-right: Suppose $\Gamma, u, \tau \vDash \psi_1 \wedge \psi_2$, i.e. $\Gamma, u, \tau \vDash \psi_1$ and $\Gamma, u, \tau \vDash \psi_2$. By the inductive hypothesis, $\Gamma, u, \tau \vDash \psi_1$ implies that there is a set Π_1 of paths s.t. $\Gamma, u, \Pi_1, \tau \vDash \psi_1$, and $\Gamma, u, \tau \vDash \psi_2$ implies that there is a set Π_2 of paths s.t. $\Gamma, u, \Pi_2, \tau \vDash \psi_2$. Set $\Pi := \Pi_1 \cup \Pi_2$. Then $\Gamma, u, \Pi, \tau \vDash \psi_1 \wedge \psi_2$.
  Right-to-left: Suppose $\Gamma, u, \Pi, \tau \vDash \psi_1 \wedge \psi_2$, i.e. there are sets Π_1, Π_2 of paths with $\Pi_1 \cup \Pi_2 = \Pi$ s.t. $\Gamma, u, \Pi_1, \tau \vDash \psi_1$ and $\Gamma, u, \Pi_2, \tau \vDash \psi_2$. By the inductive hypothesis $\Gamma, u, \tau \vDash \psi_1$ and $\Gamma, u, \tau \vDash \psi_2$.
- $\phi = \psi_1 \vee \psi_2$: Similar to the case $\phi = \psi_1 \wedge \psi_2$
- $\phi = \langle\alpha_i\rangle\psi$:
  Left-to-right: Suppose $\Gamma, u, \tau \vDash \langle\alpha_i\rangle\psi$, i.e. there is a $u' \in \mathcal{U}$ s.t. $(u, u') \in \alpha_i$ and $\Gamma, u', \tau \vDash \psi$. By the inductive hypothesis, there is a set Π' of paths s.t. $\Gamma, u', \Pi', \tau \vDash \psi$. Set $\Pi := \{(u, u') \circ \pi \mid \pi \in \Pi'\}$. Then $\Pi' = \Pi[1:]$ and $\forall \pi \in \Pi\ \pi[0] = (u, u')$. So $\Gamma, u, \Pi, \tau \vDash \langle\alpha_i\rangle\psi$.
  Right-to-left: Suppose $\Gamma, u, \Pi, \tau \vDash \langle\alpha_i\rangle\psi$, i.e. there is a $u' \in \mathcal{U}$ s.t. $\Gamma, u', \Pi[1:], \tau \vDash \psi$, $(u, u') \in \alpha_i$. By the inductive hypothesis $\Gamma, u, \tau \vDash \langle\alpha_i\rangle\psi$.
- $\phi = @_n\psi$:
  Left-to-right: Suppose $\Gamma, u, \tau \vDash @_n\psi$, i.e. $\Gamma, u', \tau \vDash \psi$, where $W(n) = u'$. By the inductive hypothesis, there is a set Π of paths s.t. $\Gamma, u', \Pi, \tau \vDash \psi$. Then $\Gamma, u, \Pi, \tau \vDash @_n\psi$.
  Right-to-left: Suppose $\Gamma, u, \Pi, \tau \vDash @_n\psi$, i.e. Γ, u', $\Pi, \tau \vDash \psi$, where $W(n) = u'$. By the inductive hypothesis $\Gamma, u', \tau \vDash \psi$, and therefore $\Gamma, u, \tau \vDash @_n\psi$.
- $\phi = @_x\psi$: Similar to the case $@_n\psi$
- $\phi = \downarrow_x \psi$:
  Left-to-right: Suppose $\Gamma, u, \tau \vDash \downarrow_x \psi$, i.e. Γ, u, $\tau[x \mapsto u] \vDash \psi$. By the inductive hypothesis, Γ, u, Π, $\tau[x \mapsto u] \vDash \psi$, i.e. $\Gamma, u, \Pi, \tau \vDash \downarrow_x \psi$.
  Right-to-left: Suppose $\Gamma, u, \Pi, \tau \vDash \downarrow_x \psi$, i.e. Γ, u, Π, $\tau[x \mapsto u] \vDash \psi$. By the inductive hypothesis, Γ, u, $\tau[x \mapsto u] \vDash \psi$, i.e. $\Gamma, u, \tau \vDash \downarrow_x \psi$.

B. PROOF OF THEOREM 2

First, we need to prove the following lemma:

LEMMA 1. *Let $\Gamma = (\mathcal{G}, W, V)$ be a model, τ a valuation and Π a set of paths in $\mathcal{U}$. Let $X_1, X_2 \in \{\text{LO}, \text{GL}\}$ and $Y_1, Y_2 \in \{\text{LI}, \text{GE}\}$ be s.t. $(X_1, Y_1, \text{W}) \leq (X_2, Y_2, \text{W})$ in the blacklist-restriction lattice. Then $Valid_{(X_2,Y_2)}(\Gamma, \Pi, \tau)$ implies $Valid_{(X_1,Y_1)}(\Gamma, \Pi, \tau)$.*

PROOF. Suppose $Valid_{(X_2,Y_2)}(\Gamma, \Pi, \tau)$. We want to show that $Valid_{(X_1,Y_1)}(\Gamma, \Pi, \tau)$. For this we have to show that the four conditions from Definition 2 are satisfied for X_1, Y_1. We call the first two conditions the globality conditions and the other two the generality conditions.

Since $(X_1, Y_1, \text{W}) \leq (X_2, Y_2, \text{W})$, we know that it is not the case that $X_1 = \text{GL}$ and $X_2 = \text{LO}$. If $X_1 = X_2$, the globality conditions are satisfied for X_1 since they are satisfied for X_2. So all we have to show is that the globality conditions are satisfied for X_1 if $X_1 = \text{LO}$ and $X_2 = \text{GL}$. Of course, since $X_1 \neq \text{GL}$, the second globality condition is trivially satisfied. Since $X_2 = \text{GL}$, we have that for all $u, u' \in \mathcal{U}$ s.t. (u, u') is an element of some $\pi \in \Pi$, $(u, u') \notin b$. So in particular, for every $u \in \mathcal{U}$ s.t. $(\tau(\mathsf{own}), u)$ is an element of

some $\pi \in \Pi$, $(\tau(\mathsf{own}), u) \notin b$. Therefore, the first globality condition is satisfied for X_1.

Similarly, it is enough to show that the first generality condition is satisfied for $Y_1 = \text{LI}$ and $Y_2 = \text{GE}$. But since $Y_2 = \text{GE}$, we know by the second generality condition for Y_2 that $(\tau(\mathsf{own}), \tau(\mathsf{req})) \notin b$, so that the first generality condition for Y_1 is satisfied. □

We now proceed to proving Theorem 2. Let $X_1, X_2 \in \{\text{LO}, \text{GL}\}$, $Y_1, Y_2 \in \{\text{LI}, \text{GE}\}$ and $Z_1, Z_2 \in \{\text{W}, \text{S}\}$ be s.t. $(X_1, Y_1, Z_1) \leq (X_2, Y_2, Z_2)$ in the blacklist-restriction lattice. Suppose $\mathcal{I}, u, \tau \vDash \phi_{(X_2,Y_2,Z_2)}$. We need to show that $\mathcal{I}, u, \tau \vDash \phi_{(X_1,Y_1,Z_1)}$. Since $(X_1, Y_1, Z_1) \leq (X_2, Y_2, Z_2)$, we know that it is not the case that $Z_1 = \text{S}$ and $Z_2 = \text{W}$. We consider the other three possible values for Z_1, Z_2 separately:

- $Z_1 = Z_2 = \text{W}$:
  Since we have $Z_2 = \text{W}$, $\mathcal{I}, u, \tau \vDash \phi_{(X_2,Y_2,Z_2)}$ implies that there is a set Π of paths s.t. $\mathcal{I}, u, \Pi, \tau \vDash \phi$ and $\mathit{Valid}_{(X_2,Y_2)}(\Pi)$. Since $(X_1, Y_1, \text{W}) \leq (X_2, Y_2, \text{W})$, Lemma 1 implies that $\mathit{Valid}_{(X_1,Y_1)}(\Pi)$. Hence, $\mathcal{I}, u, \tau \vDash \phi_{(X_1,Y_1,Z_1)}$.
- $Z_1 = Z_2 = \text{S}$:
  In this case, $\mathcal{I}, u, \tau \vDash \phi_{(X_2,Y_2,Z_2)}$ implies that (i) there is a set Π of paths s.t. $\mathcal{I}, u, \Pi, \tau \vDash \phi$, and (ii) for every set Π of paths s.t. $\mathcal{I}, u, \Pi, \tau \vDash \phi$, $\mathit{Valid}_{(X_2,Y_2)}(\Pi)$. For showing that $\mathcal{I}, u, \tau \vDash \phi_{(X_1,Y_1,Z_1)}$, it is enough to show that for every set Π of paths s.t. $\mathcal{I}, u, \Pi, \tau \vDash \phi$, $\mathit{Valid}_{(X_1,Y_1)}(\Pi)$. So let Π be a set of paths s.t. $\mathcal{I}, u, \Pi, \tau \vDash \phi$. It is now enough to show that $\mathcal{I}, u, \Pi, \tau \vDash \phi$, $\mathit{Valid}_{(X_1,Y_1)}(\Pi)$. By (ii), $\mathit{Valid}_{(X_2,Y_2)}(\Pi)$. $(X_1, Y_1, \text{S}) \leq (X_2, Y_2, \text{S})$ implies that $(X_1, Y_1, \text{W}) \leq (X_2, Y_2, \text{W})$. This together with Lemma 1 implies that $\mathit{Valid}_{(X_1,Y_1)}(\Pi)$, as required.
- $Z_1 = \text{W}$ and $Z_2 = \text{S}$:
  Since $Z_2 = \text{S}$, $\mathcal{I}, u, \tau \vDash \phi_{(X_2,Y_2,Z_2)}$ implies that (i) there is a set Π of paths s.t. $\mathcal{I}, u, \Pi, \tau \vDash \phi$, and (ii) for every set Π of paths s.t. $\mathcal{I}, u, \Pi, \tau \vDash \phi$, $\mathit{Valid}_{(X_2,Y_2)}(\Pi)$. (i) and (ii) together imply that there is a set Π of paths s.t. $\mathcal{I}, u, \Pi, \tau \vDash \phi$ and $\mathit{Valid}_{(X_2,Y_2)}(\Pi)$. $(X_1, Y_1, \text{W}) \leq (X_2, Y_2, \text{S})$ implies that $(X_1, Y_1, \text{W}) \leq (X_2, Y_2, \text{W})$. This together with Lemma 1 implies that $\mathit{Valid}_{(X_1,Y_1)}(\Pi)$. Hence, $\mathcal{I}, u, \tau \vDash \phi_{(X_1,Y_1,Z_1)}$.

C. SKETCH OF PROOF OF THEOREM 3

First note by inspection of the definition of the path semantics that a path in a set of paths satisfying a formula ϕ corresponds to a branch in the syntax tree of ϕ starting at the root (which is labeled by ϕ) and ending in a node labeled by a strictly positive subformula of ϕ of the form x, m, p or $\neg\psi$. The edges in this branch that connect a node labeled $\langle\alpha_i\rangle\psi$ to ψ correspond to the edges of the path.

Note that in Definition 2, where we defined which sets of paths are free of blacklist problems, the first, second and fourth condition actually refer to the set Π of paths, whereas the third condition does not refer to this set and hence is independent of the choice of Π. For this reason, Algorithm 2 handles the restrictions imposed by this condition somewhat differently from the restrictions imposed by the other three conditions. To refer to the restrictions imposed by the other three conditions, we define $v_{(X,Y)}(\Gamma, \Pi, \tau)$ as follows:

Definition 6. Let $\mathcal{I} = (\mathcal{G}, W, V)$ be a model and τ be a valuation. For $X \in \{\text{LO}, \text{GL}\}$, $Y \in \{\text{LI}, \text{GE}\}$ and a set Π of paths, we define $v_{(X,Y)}(\mathcal{I}, \Pi, \tau)$ to hold iff the following four properties are satisfied:

- If $X = \text{LO}$, then for every $u \in \mathcal{U}$ s.t. $(\tau(\mathsf{own}), u)$ is an element of some $\pi \in \Pi$, $(\tau(\mathsf{own}), u) \notin b$.
- If $X = \text{GL}$, then for all $u, u' \in \mathcal{U}$ s.t. (u, u') is an element of some $\pi \in \Pi$, $(u, u') \notin b$.
- If $Y = \text{GE}$, then $(\tau(\mathsf{own}), \tau(\mathsf{req})) \notin b$, and for all $u, u' \in \mathcal{U}$ s.t. (u, u') is an element of some $\pi \in \Pi$, $(\tau(\mathsf{own}), u) \notin b$ and $(\tau(\mathsf{own}), u') \notin b$.

We first sketch how to prove the theorem for $Z = \text{W}$: The insertion of $\downarrow_{x_k}$'s and $\downarrow_{y_k}$'s into ϕ in line 3 of algorithm does not affect which sets of paths satisfy ϕ, but makes it possible to refer to the nodes of these paths. The new subformulas, which in lines 6-11 of Algorithm 2 get conjuncted to strictly positive subformulas χ of ϕ of the form x, m, p or $\neg\psi$, make use of this possibility to refer to the nodes of the paths in order to express the conditions for $v_{(X,Y)}(\Gamma, \Pi, \tau)$ within ϕ. In line 12 we ensure that if $Y = \text{LI}$, then $(\tau(\mathsf{own}), \tau(\mathsf{req})) \notin b$. Hence the modifications performed on ϕ in case $Z = \text{W}$ ensure that $\phi[X, Y, Z]$ is satisfied precisely by those sets of paths Π that satisfy ϕ and $\mathit{Valid}_{(X,Y)}(\Gamma, \Pi, \tau)$, i.e. precisely by those sets of paths that satisfy $\phi_{(X,Y,Z)}$.

Now we sketch how to prove the theorem for $Z = \text{S}$: Note that for a set Π of paths to satisfy a formula ϕ of the form $\psi_1 \vee \psi_2$, it is enough that it satisfies ψ_1 or ψ_2. Hence, concerning the correspondence mentioned in the first paragraph of this proof sketch, only the branches of the syntax tree of one of ψ_1 and ψ_2 correspond to paths in Π, while the branches in the syntax tree of the other are not reflected in the structure of Π at all. In general, we can say that the correspondence is only a one-to-one correspondence, if ϕ is a formula that does not have a strictly positive subformula of the form $\psi_1 \vee \psi_2$. This is why for the case $Z = \text{S}$, Algorithm 2 makes use of the Disjunctive Form $DF(\phi)$ of ϕ: The modifications made to the disjuncts ϕ_i depend on the correspondence between paths and branches of the syntax tree being one-to-one.

Furthermore, note that one can easily prove by an induction over the length of ϕ that every hybrid logic formula ϕ is equivalent to its Disjunctive Form.

In lines 18-23 of Algorithm 2, we define – for each strictly positive subformula $\chi_{i,j}$ of ϕ_i of the form x, m, p of $\neg\psi$ – a formula $\psi_{i,j}$ that expresses that the conditions for $v_{(X,Y)}(\Gamma, \{\pi\}, \tau)$ are not satisfied, where π is the path corresponding to the syntax tree branch ending at $\chi_{i,j}$. Hence, $\phi_{i,j}$ as defined in line 24 has the following property: $\mathcal{I}, u, \tau \vDash \phi_{i,j}$ iff there is a set Π of paths s.t. $\mathcal{I}, u, \Pi, \tau \vDash \phi_i$ and it is not the case that $v_{(X,Y)}(\Gamma, \{\pi\}, \tau)$ (where $\pi \in \Pi$ is the path corresponding to the syntax tree branch ending at $\chi_{i,j}$). This implies that $\mathcal{I}, u, \tau \vDash \neg\phi_{i,j}$ iff for every set Π of paths s.t. $\mathcal{I}, u, \Pi, \tau \vDash \phi_i$, we have $v_{(X,Y)}(\Gamma, \{\pi\}, \tau)$. Hence, $\overline{\phi_i}$ as defined in line 25 has the following property: $\mathcal{I}, u, \tau \vDash \overline{\phi_i}$ iff for every set Π of paths with $\mathcal{I}, u, \Pi, \tau \vDash \phi_i$, $v_{(X,Y)}(\Gamma, \{\pi\}, \tau)$ holds for every path in $\pi \in \Pi$, i.e. $v_{(X,Y)}(\Gamma, \Pi, \tau)$.

Now the equivalence between ϕ and $DF(\phi)$ together with the property of $\overline{\phi_i}$ that we just established implies the following property for the $\overline{\phi}$ defined in line 26: $\mathcal{I}, u, \tau \vDash \overline{\phi}$ iff $\mathcal{I}, u, \tau \vDash \phi$ and for every set Π of paths with $\mathcal{I}, u, \Pi, \tau \vDash \phi$, we have $v_{(X,Y)}(\Gamma, \Pi, \tau)$. Concerning the ψ defined in line 27, this implies that $\mathcal{I}, u, \tau \vDash \psi$ iff $\mathcal{I}, u, \tau \vDash \phi$ and for every set Π of paths with $\mathcal{I}, u, \Pi, \tau \vDash \phi$, we have $\mathit{Valid}_{(X,Y)}(\Gamma, \Pi, \tau)$, i.e. $\mathcal{I}, u, \tau \vDash \psi$ iff $\mathcal{I}, u, \tau \vDash \phi_{(X,Z,Y)}$, as required.