

# Social Access Control Language (SocACL)

Edward Caprin and Yan Zhang  
School of Computing, Engineering and  
Mathematics.  
University of Western Sydney, Kingswood,  
Australia.  
[e.caprin][y.zhang]@uws.edu.au

Khaled M. Khan  
Department of Computer Science and  
Engineering.  
Qatar University, Qatar.  
k.khan@qu.edu.qa

## ABSTRACT

*Online Social Networks* hold vast amounts of readily accessible personal information leaving them particularly vulnerable to privacy breach attacks [6]. With the impact these breaches varying from simply embarrassing the user, to negatively influencing the decision of potential employers, identity theft and even physical harm it is important that they are addressed. In this research we approach privacy management in OSNs as an access control problem. We propose a formal *Attribute-Based Access Control* (ABAC) language; *SocACL*. SocACL is based on *Answer Set Programming* (ASP) and allows for policy specification using the most abundant sources of information available in OSNs; user attributes and relationships. This paper outlines SocACL's core concepts, features, syntax and semantics.

**Categories and Subject Descriptors:** C.2.0 [Computer-Communication Networks]: General - *Security and Protection*. D.4.6 [Operating Systems]: Security and protection - *Access controls*. I.2.3 [Artificial Intelligence]: Deduction and Theorem Proving - *Logic programming*.

**General Terms:** Security, Languages.

**Keywords:** Answer Set Programming, Online Social Networks, Privacy, Attribute-Based Access Control.

## 1. INTRODUCTION

*Online Social Networks* (OSNs), such as Facebook, encourage their users to disclose significant amounts of personal information to facilitate connecting and sharing with others. This has resulted in some OSNs holding vast amounts of information about their users; all of which is readily available via their profile page. As a result, OSNs are particularly vulnerable to privacy breach attacks [6]. With the impact these breaches varying from simply embarrassing the user, to negatively influencing the decision of potential employers, identity theft and even physical harm it is important that they are addressed. OSN operators have responded to growing privacy concern by providing their users with customis-

able privacy settings. However, these have, in general, been ineffective and often result in settings that do not reflect the user's intentions [9]. In part this is due to the coarse-grained nature of the information on which these settings are based and the lack of tools for the user to verify the correctness of these settings [8].

In this research we approach privacy management in OSNs as an access control problem. We propose a fine-grained, formal *Attribute-Based Access Control* (ABAC) language called *SocACL* (Social Access Control Language). SocACL is based on *Answer Set Programming* (ASP) and utilises model checking for policy evaluation.

This paper serves as a semantic foundation to SocACL beginning with related work in section 2. Section 3 outlines the syntax and semantics of SocACL. While in section 4 we discuss queries. We conclude in section 5 with comments on future work.

## 2. RELATED WORK

Relationships play an integral role in any OSN, as such there have been numerous access control frameworks proposed based on them [3, 5]. Dhia [3] propose a framework based on node reachability by constructing a *social graph* from relationships and user profile pages. By mapping permissions to sequences of relationships [3] reduce policy evaluation to finding paths in the social graph. While *Relationship-Based Access Control* (ReBAC) [5] describes access permissions in terms of the accessors relationship with the owner. These relationships can be inverted to derive the opposite direction, e.g. inverse of a "parent" relationship is "child". Furthermore, primitive relationships, such as "parent", can be combined to form complex ones, "parent parent" is a "grandparent".

In [3, 5] relationships are mutually agreed upon bidirectional social links between two principals. In turn, making them a shared resource of the two principals. With shared resources in OSNs presenting their own issues [7] SocACL takes a different approach. SocACL relationships are unidirectional and are treated as an attribute of a principal denoting the relationship the believe they have with another.

Besides relationship based approaches, there has been a trend towards more context centric models such as ABAC for OSNs and other open web services [2, 10]. However, ABAC itself is problematic as it requires the combining of a wide array of potentially conflicting sources of security relevant information into a coherent framework. As explained by [2], this often results in ABAC languages being forced to make a trade off between having either a rich set of features

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SIN'13, November 26 - 28, 2013, Aksaray, Turkey  
Copyright 2013 ACM 978-1-4503-2498-4/13/11 ...\$15.00.

or well defined semantics. Crampton et al. [2] address this by separating policy target and permission specification into two distinct problems, defining sub-languages for each. SocACL overcomes the need for this trade off by defining its semantics as a translation to ASP.

*Answer Set Programming* (ASP) is a form of declarative programming based on the *stable model semantics* [1]. It is well suited to representing domain specific knowledge [1], such as user information in OSNs. Yuan and Tong [10] suggest the evaluation of first order logic or variant expressions, such as ASP, for ABAC policy evaluation. Our translation allows us to do this. By translating a SocACL policy base to a representative ASP program we are able to utilise inference engines, such as DLV [4], as the basis of the language’s policy evaluation system.

### 3. SOCACL

$$Prin \text{ says } Head \text{ if } Body; \quad (1)$$

$$Head'_{Prin} \leftarrow Body'. \quad (2)$$

SocACL *Policy Bases* (PBs) are a set of statements which describes a principals’ attributes, privacy preferences and opinions of others. These statements, generally, take the form (1). Where principal *Prin* is making a statement on *Head*, which can be an attribute (section 3.1), relationship (section 3.2) or authorisation (section 3.5). This statement is true if and only if *Body* is true. *Body* is a conjunction of decision criteria which can include attributes, relationships, constraints on variables, aggregates (section 3.3) and descriptions (section 3.4). We define the semantics of (1) by its transformation to the ASP rule shown in (2).  $Head'_{Prin}$  is a *Head* translated w.r.t. *Prin*, which we clarify in later sections.  $Body'$  is all the decision criterion in *Body* translated.

Figure 1 shows an EBNF of SocACL where NAME starts with a lowercase letter and can contain letters, numbers and underscores, while VAR starts with an uppercase letter. OBN, RCN and DN are, respectively, the obligation, relationship chain and description name.

#### 3.1 Attributes

$$Prin \text{ says } P \cdot Attr \cdot Fields : S \cdot Pr \text{ if } Body; \quad (3)$$

$$Attr(Prin, P, Fields', S, Pr) \leftarrow Body'. \quad (4)$$

Attributes are facts about a principal which can have values associated with it, e.g. hair colour has the value “brown”. In SocACL, principals make statements about principals’ attributes and infer attributes from other information using the form (3). *Prin* states principal *P* has some attribute called *Attr*.  $Fields = f_1 \cdot \dots \cdot f_n$ , where  $f_i$  is some value associated with this attribute. When  $n = 0$  the attribute has no associated values, allowing  $\cdot Fields$  to be omitted, e.g. `alice says alice · married : ns · np`.  $S \in \{s, ns\}$  is the *Sensitivity Flag* (SF), indicating if the attribute is sensitive or not.  $Pr \in \{p, np\}$  is the *Primary Instance Flag* (PIF), denoting whether this is the primary instance of a this attribute, e.g. an individual may have many phone numbers, but one is their primary number. Both these flags are used during the SocACL negotiation process which is not covered

Query	=	NAME ‘asks’ NAME · ACT · OBJ · PU;’
Policy	=	{NAME ‘says’ (Rule   Definition) ‘;’}
Rule	=	Head [‘if’ Body]
Definition	=	Def-Obli   Def-RelC   Def-Desc
Head	=	Auth   Attr ‘:’ SF · PIF   Rel-Dir ‘:’ SF
Body	=	( BTerm   Aggr   Cons ) [ ‘,’ Body ]
BTerm	=	[ ‘not’ ] [ PRIN ‘says’ ] ( Attr   Desc   Rel )
Auth	=	( ‘allow’   ‘deny’ ) · PRIN · ACT · OBJ · PU · OBN
Attr	=	Prin · ATTR-NAME [ { · Val } ]
Def-Obli	=	‘define’ · ‘obligation’ · OBN · ACT · Prin
Def-RelC	=	‘define’ · ‘relchain’ · RCN · ( ‘Body’ )
Def-Desc	=	‘define’ · ‘description’ · DN · VAR · ( ‘Body’ )
Aggr	=	VAR ‘=’ Aggr-Op · VAR · ( ‘Body’ )
Aggr-Cmp	=	( ‘exactly’   ‘atleast’   ‘atmost’ ) · Val
Aggr-Op	=	( ‘count’   ‘sum’   ‘min’   ‘max’ )
Desc	=	SUB · ‘description’ · DN
Rel	=	Rel-Dir   Rel-Sind   Rel-Rind
Rel-Dir	=	SUB · ‘relationship’ · REL-TYPE · SUB
Rel-Sind	=	SUB · ‘sindRelationship’ · RCN · SUB
Rel-Rind	=	SUB · ‘rindRelationship’ · NUM · SUB
Cons	=	Val ( ‘<’   ‘>’   ‘≤’   ‘≥’   ‘=’   ‘≠’ ) Val
Prin	=	SUB   OBJ
Val	=	NAME   VAR   NUM

Figure 1: EBNF of SocACL.

in this paper. Equation (4) is the translation, and thus the semantics, of (3), where  $Fields' = f_1, \dots, f_n$ .

For the rest of this paper we use the running example of a principal Alice in a hypothetical OSN. Alice has organised her uploaded content in the folders shown in Figure 2, where “public” and “private” are sub-folders of “gallery”, and so on.

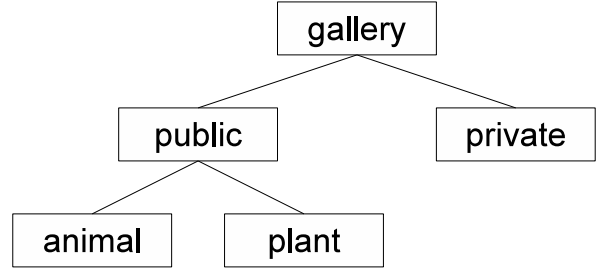


Figure 2: Alice’s Gallery Folders

*Example 1.* Figure 2 can be represented using attributes where an object *A* has an attribute representing which folder it is in.

`alice says A · isIn · public : ns · np if A · isIn · animal;`  
`alice says A · isIn · public : ns · np if A · isIn · plant;`

The above SocACL statements denote the folder “animal” and “plant” are sub-folders of “public”. This translates to:

$isIn(alice, A, public, ns, np) \leftarrow isIn(\_, A, animal, \_, \_).$   
 $isIn(alice, A, public, ns, np) \leftarrow isIn(\_, A, plant, \_, \_).$

We see as a result of the translation, the principal making the statement has become the first arity (alice), while the principal the attribute relates to is the second (*A*), followed by the attributes values (public,animal,plant,etc). For attributes the last two arities are always the SF and PIF. When attributes are used in *Body* the form (5) is used.

$$P \cdot \text{Attr} \cdot \text{Fields} \quad (5)$$

$$\text{Attr}(\text{Prin}, P, \text{Fields}', -, -) \quad (6)$$

Where (5) and its translation (6) are consistent with (3) and (4) respectively. Referring to Figure 1, attributes used in *Body* can have *Prin says* proceed them. This restricts which principals the policy specifier trusts on that attribute, acting as a form of delegation. When *Prin says* is not present in *Body*, *Prin* is substituted by an underscore during translation. Also, in (6) the last two arities, the SF and PIF, are underscores, the anonymous variable of DLV.

### 3.2 Relationships

SocACL provides three different types of relationships; *Direct*, *Strict-Indirect* and *Relaxed-Indirect*. For the continuation of this paper our examples will be in consideration to the social graph shown in Figure 3.

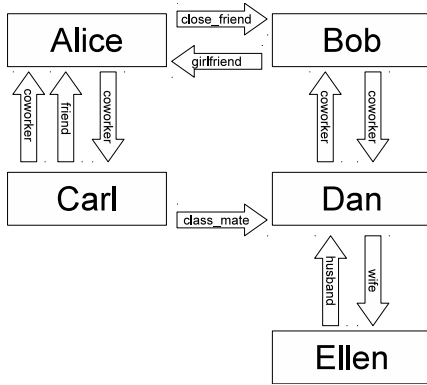


Figure 3: Social Graph.

*Prin says*  $P \cdot \text{relationship} \cdot rt \cdot \text{Sub} : S$  if *Body*; (7)

$$\text{relationship}(\text{Prin}, P, \text{Sub}, rt, S) \leftarrow \text{Body}', P \neq \text{Sub}. \quad (8)$$

Direct relationships (7) are an attribute denoting a 1st-degree relationship a principal believes they hold with another, which may or may not be mirrored. Similar to attributes, direct relationships can be inferred from other information. In (7) *Prin* is stating principal *P* believes they have a 1st-degree relationship of the type *rt* with principal *Sub*. Its semantics, given in (8), prevents a principal from being in a relationship with itself.

*Example 2.* Alice considers Bob a “close-friend”, while Bob considers Alice his “girlfriend”:

alice **says** alice · relationship · close\_friend · bob : ns;

bob **says** bob · relationship · girlfriend · alice : s;

Translation of the above would result in:

**relationship**(alice, alice, bob, close\_friend, ns).

**relationship**(bob, bob, alice, girlfriend, s).

In this case both Alice and Bob agree that there is a relationship between them, but they do not agree on its extent. Similar to attributes, relationships take on a different form when used in *Body*. Besides direct relationships, *Body* can contain indirect relationships, such as “friend-of-a-friend”.

$$P \cdot \text{relationship} \cdot rt \cdot \text{Sub} \quad (9)$$

$$P \cdot \text{strictRelationship} \cdot \text{RCN} \cdot \text{Sub} \quad (10)$$

$$P \cdot \text{relaxedIndirectRelationship} \cdot \text{Depth} \cdot \text{Sub} \quad (11)$$

In the above we have the various *Body* forms for relationships, where (9), (10) and (11) are direct, strict-indirect (sind) and relaxed-indirect (rind) relationships, respectively. Below, (12), (13) and (14) show their respective translations.

$$\text{relationship}(\text{Prin}, P, \text{Sub}, rt, -) \quad (12)$$

$$\text{strictRelationship}(\text{Prin}, P, \text{Sub}, \text{RCN}, -) \quad (13)$$

$$\text{relaxedIndirectRelationship}(\text{Prin}, P, \text{Sub}, \text{Depth}, -) \quad (14)$$

Sind relationships are indirect relationships expressed as a sequence of direct relationships, which is defined using a *Relationship Chain Definition*, shown in (15). *RCN* is the *Relationship Chain Name* and  $rt_i, 1 \leq i \leq n$ , is the relationship type that must exist at the  $i^{\text{th}}$ -degree of separation. Equation (16) shows the definition’s semantics. The intuition is that the sind relationship holds when each direct relationship exists at each intermediary principal, and all of these principals are unique. The principal uniqueness is to prevent unexpected access control decisions resulting from relationship “backtracking”.

$$\text{Prin says define} \cdot \text{relchain} \cdot \text{RCN} \cdot (rt_1, \dots, rt_n); \quad (15)$$

$$\text{strictRelationship}(\text{Prin}, P, \text{Sub}_n, \text{RCN}, s) \leftarrow$$

$$\text{relationship}(\text{Prin}, P, \text{Sub}_1, rt_1, -), \dots, \quad (16)$$

$$\text{relationship}(\text{Sub}_{n-1}, \text{Sub}_{n-1}, \text{Sub}_n, rt_n, -),$$

$$P \neq \text{Sub}_1, \dots, \text{Sub}_{n-1} \neq \text{Sub}_n.$$

Rind relationships define an indirect relationship in terms of distance (*Depth*) between two principals’ and are calculated by the below ASP rules which are included as part of every translated SocACL PB. They find the shortest path between two nodes in a graph, while also ensuring the participating principals are unique. For both sind and rind relationships evaluation is similar to the node reachability checks of [3].

$$\text{path}(X, Y, 1) \leftarrow \text{relationship}(X, X, Y, R, -).$$

$$\text{path}(X, Z, D) \leftarrow \text{path}(X, Y, D1), \text{path}(Y, Z, 1),$$

$$+ (D1, 1, D), X \neq Y, Y \neq Z, X \neq Z.$$

$$\text{relaxedIndirectRelationship}(X, X, Y, D, s) \leftarrow \text{path}(X, Y, -),$$

$$D = \# \min\{D1 : \text{path}(X, Y, D1)\}.$$

### 3.3 Aggregates

$$Aggr \cdot (Tar) \cdot (Body) \cdot ACmp \cdot LB \cdot UB \quad (17)$$

$$V = Aggr \cdot (Tar) \cdot (Body) \quad (18)$$

SocACL supports the use of the aggregate operations count, sum, min and max. These can be applied by either comparing the operation result to other values to get a boolean answer or assign the result to a variable. In (17),  $Aggr \in \{\mathbf{count}, \mathbf{sum}, \mathbf{min}, \mathbf{max}\}$  is the aggregate operation applied to  $Tar$  in  $Body$ . The result of this operation is then compared with number  $LB$  and  $UB$  depending on  $ACmp \in \{\mathbf{exactly}, \mathbf{atleast}, \mathbf{atmost}, \mathbf{between}\}$ . Note that  $UB$  is only needed for **between**. Alternatively, using (18) the result can be assigned to some variable  $V$ .

$$L \# Aggr\{Tar : Body'\} U, Body'_{Tar} \quad (19)$$

$$V = \# Aggr\{Tar : Body'\}, Body'_{Tar} \quad (20)$$

Equation (19) is the translation of (17), while (20) is the translation of (18).  $L$  and  $U$  are substituted depending on  $ACmp$  from (17). When  $ACmp = \mathbf{exactly}$ ;  $L$  is omitted and  $U = "= LB"$ , minus the quotations.  $ACmp = \mathbf{atleast}$ ;  $L = "LB \leq "$  and  $U$  is omitted.  $ACmp = \mathbf{atmost}$ ;  $L$  is omitted and  $U = "\leq LB"$ .  $ACmp = \mathbf{between}$ ;  $L = "LB \leq "$  and  $U = "\leq UB"$ .  $Body'_{Tar}$  is a special translation of  $Body$  where instances of  $Tar$  are replaced with  $Tar1$ ,  $Tar$  followed by the number 1. This is to accommodate DLV's aggregate implementation where  $Tar$  cannot occur outside of the curly braces.

### 3.4 Descriptions

$$Prin \text{ says define} \cdot \mathbf{description} \cdot DN \cdot P \cdot (Body); \quad (21)$$

$$\mathbf{description}(Prin, P, DN) \leftarrow Body'. \quad (22)$$

These allow the policy specifier to group decision criteria to form a description of a principal, creating a reusable  $Body$ . Similar sind relationships, descriptions need to be defined using the form (21), translation (22).  $DN$  is the *Description Name* and  $P$  is a principal that occurs in  $Body$ . Once a description is defined it is then referenced by its name in  $Body$  using the (23), translation (24).

$$P \cdot \mathbf{description} \cdot DN \quad (23)$$

$$\mathbf{description}(Prin, P, DN) \quad (24)$$

*Example 3.* Alice describes an Object as a "animalPhoto" if it is in the folder "animal" and is of the type "photo" is shown below.

$$\text{alice says define} \cdot \mathbf{description} \cdot \mathbf{animalPhoto} \cdot \mathbf{Object} \cdot (\mathbf{Object} \cdot \mathbf{isIn} \cdot \mathbf{animal}, \mathbf{Object} \cdot \mathbf{type} \cdot \mathbf{photo});$$

Which translates to:

$$\mathbf{description}(\text{alice}, \mathbf{Object}, \mathbf{animalPhoto}) \leftarrow \mathbf{isIn}(\_, \mathbf{Object}, \mathbf{animal}, \_, \_), \mathbf{type}(\_, \mathbf{Object}, \mathbf{photo}, \_, \_).$$

### 3.5 Authorisations

$$Prin \text{ says } Perm \cdot P \cdot Act \cdot Obj \cdot Pu \cdot ObN \text{ if } Body; \quad (25)$$

$$Perm(Prin, P, Act, Obj, Pu, ObN) \leftarrow Body', \quad (26)$$

$$\mathbf{acceptOb}(P, Prin, ObN).$$

SocACL provides positive and negative authorisations with deny override behaviour, taking the form (25).  $Perm \in \{\mathbf{allow}, \mathbf{deny}\}$  is the permission type, allowing or denying principal  $P$  from performing action  $Act$  on object  $Obj$  for the purpose  $Pu$ .  $ObN$  identifies the obligation this authorisation is associated with, in the case of no obligation;  $ObN = \mathbf{none}$ . An obligation is some action to be performed after  $P$  has acted on this permission. As seen by its translation in (26), to be granted an authorisation,  $P$  must accept the obligation and all other criteria in  $Body$  must be met. Authorisations are then mapped to actions by the below ASP rule which forms part of every translated SocACL PB.

$$\mathbf{action}(P, Prin, Act, Obj, Pu) \leftarrow \mathbf{allow}(Prin, P, Act, Obj, Pu, ObN), \mathbf{not} \ \mathbf{deny}(Prin, P, Act, Obj, Pu, ObN).$$

*Example 4.* Alice wants to allow people in at least a 2nd-degree relationship with her to view Objects that fit the "animalPhoto" description for social purposes is shown below, followed by its translation.

$$\text{alice says allow} \cdot \mathbf{Other} \cdot \mathbf{view} \cdot \mathbf{Object} \cdot \mathbf{social} \cdot \mathbf{none} \text{ if}$$

$$\text{alice} \cdot \mathbf{rindRelationship} \cdot A \cdot \mathbf{Other}, A \leq 2,$$

$$\mathbf{Object} \cdot \mathbf{description} \cdot \mathbf{animalPhoto};$$

$$\mathbf{allow}(\text{alice}, \mathbf{Other}, \mathbf{view}, \mathbf{Object}, \mathbf{social}, \mathbf{none}) \leftarrow$$

$$\mathbf{rindRelationship}(\text{alice}, \text{alice}, \mathbf{Other}, A, \_), A \leq 2$$

$$\mathbf{description}(\text{alice}, \mathbf{Object}, \mathbf{animalPhoto}).$$

As previously mentioned, obligations define some action to be performed in response to acting on some permission. For instance, before a parent agrees to give their child an ice cream the child agrees to clean their room. In order to use an obligation one must first define one using the form (27), translation (28).

$$Prin \text{ says define} \cdot \mathbf{obligation} \cdot ObN \cdot OAct \cdot Tar; \quad (27)$$

$$\mathbf{obligation}(Prin, ObN, OAct, Tar). \quad (28)$$

$ObN$  is the obligation name.  $OAct$  is the action to be performed to fulfill the obligation and  $Tar$  is what this action is to be performed on. There is currently no feature in SocACL that allows a principal to specify when they reject an obligation and no mechanism that allows for the fulfilment of obligations. As such, both are noted as future work.

## 4. QUERIES

$$Prin \text{ asks } P \cdot Act \cdot Obj \cdot Pu; \quad (29)$$

$$\mathbf{action}(Prin, P, Act, Obj, Pu)? \quad (30)$$

A query is a request from one principal  $Prin$  to another,  $P$ , asking to perform action  $Act$  on  $Obj$  for the purpose  $Pu$ . In SocACL, queries take the form (29), translation (30). Query answering in SocACL is an assignment of truth values to a query with respect to a PB. With SocACL’s semantics defined as a translation to ASP, query answering is done through model checking using inference engines to perform the computations.

For a SocACL query  $\phi$  and PB  $\mathcal{P}$ .  $\mathcal{P} \models \phi$  iff  $\Pi \models \psi$ . Meaning that  $\mathcal{P}$  satisfies query  $\phi$ , answers yes, if and only if  $\psi$  is satisfied in every answer set of  $\Pi$ , where  $\psi$  is the translation of  $\phi$ , as shown in (30) and program  $\Pi$  is the translation of  $\mathcal{P}$ , as shown in section 3.

Programs resulting from our translation are *Normal Logic Programs* (NLPs) with arbitrary nonmonotonic negation, and may include aggregates. Since the complexity results of reasoning over these of programs are already known we reference the results of [4] who show that the complexity of this problem is co-NP when the program does not contain aggregates raises to  $\Pi_2^P$ -complete when it does.

*Example 5.* In this example we consider the PB of Alice which contains all of the SocACL statements from the previous examples in addition to attributes for the photos “cats.jpg” and “dogs.jpg”, both of which are in the “animal” folder. Running DLV with the “-filter=action” option results in the output of all actions that can be performed as a result of the authorisations in the translation of Alice’s PB.

```
{ action(bob, alice, view, "cats.jpg", social),
  action(bob, alice, view, "dogs.jpg", social),
  action(carl, alice, view, "cats.jpg", social),
  action(carl, alice, view, "dogs.jpg", social),
  action(dan, alice, view, "cats.jpg", social),
  action(dan, alice, view, "dogs.jpg", social) }
```

We see in the above answer set that Bob, Carl, and Dan can view both photos for social purposes. Ellen on the other hand cannot since she is not in at least a 2nd-degree relationship with Alice. To further demonstrate this we apply two queries to the policy, one for Carl and another for Ellen asking for permission to view “cats.jpg” for social purposes. Below we show the query for Carl followed by its translation. For Ellen, the query along with its translation would be the same except “carl” is replaced with “ellen”.

```
carl asks alice · view · “cats.jpg” · social;
action(carl,alice,view,“cats.jpg”,social)?
```

Applying both queries to the translated PB using DLV’s built in query system yields the below output. We see that Ellen cannot view the photo while Carl can. These results are based on the answer sets DLV was able to generate based on Alice’s PB. For Ellen’s query DLV concludes it is false since not all of the answer sets of Alice’s policy based can satisfy the query. Whereas, Carl’s query can be satisfied by them, and is therefore true.

```
action(carl, alice, view, "cats.jpg", social) is
  cautiously true.
action(ellen, alice, view, "cats.jpg", social) is
  cautiously false.
```

## 5. CONCLUSION AND FUTURE WORK

SocACL serves as an implementation of ABAC which integrates the privacy management concepts of purpose and

obligations into its permissions. Through its arbitrary attribute and relationship types, SocACL can accommodate the wide range of features found in various OSNs. Furthermore, with SocACL’s semantics a translation to ASP we have been able to exploit model checking for policy evaluation.

Continued work on SocACL will involve further development of obligations, obligation enforcement, and negotiation based queries to maximise privacy outcomes. Since policies at some point have to be updated to reflect the user’s ever changing privacy preferences an update framework for SocACL policies is also noted for our future work.

## 6. ACKNOWLEDGMENTS

This publication was made possible by the support of an NPRP grant (NPRP 09-079-1-013) from the Qatar National Research Fund (QNRF). The statements made herein are solely the responsibility of the authors.

## 7. REFERENCES

- [1] C. Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press, 1st edition, 2010.
- [2] J. Crampton and C. Morisset. PTaCL: A Language for Attribute-Based Access Control in Open Systems. *Lecture Notes in Computer Science*, 7215 LNCS:390–409, 2012.
- [3] I. B. Dhia. Access control in social networks: a reachability-based approach. In *Proc. of the 2012 Joint EDBT/ICDT Workshops*, EDBT-ICDT ’12, pages 227–232, New York, NY, USA, 2012. ACM.
- [4] W. Faber, G. Pfeifer, N. Leone, T. Dell’Armi, and G. Ielpa. Design and implementation of aggregate functions in the DLV system. *Theory and Practice of Logic Programming*, 8:545–580, 10 2008.
- [5] P. W. Fong. Relationship-based access control: protection model and policy language. In *Proc. of the 1st ACM Conf. on Data and Application Sec. and Pri.*, CODASPY ’11, pages 191–202, New York, NY, USA, 2011. ACM.
- [6] H. Gao, J. Hu, T. Huang, J. Wang, and Y. Chen. Security Issues in Online Social Networks. *Internet Computing, IEEE*, 15(4):56–63, July-Aug. 2011.
- [7] H. Hu, G. Ahn, and J. Jorgensen. Multiparty access control for online social networks: Model and mechanisms. *Knowledge and Data Engineering, IEEE Trans. on*, 25(7):1614–1627, 2013.
- [8] H. R. Lipford, A. Besmer, and J. Watson. Understanding Privacy Settings in Facebook with an Audience View. In *Proc. of the 1st Conf. on Usability, Psychology, and Sec.*, UPSEC’08, pages 2:1–2:8, Berkeley, CA, USA, 2008. USENIX Association.
- [9] M. Madejski, M. Johnson, and S. Bellovin. A Study of Privacy Settings Errors in an Online Social Network. In *Proc. of Pervasive Computing and Comm. Workshops (PERCOM Workshops), 2012 IEEE Int. Conf. on*, pages 340–345, March 2012.
- [10] E. Yuan and J. Tong. Attributed based access control (ABAC) for web services. In *Web Services, 2005. ICWS 2005. Proc. 2005 IEEE Int. Conf. on*. IEEE, 2005.