A Logical Approach to Privacy-Aware Access to Data

Dennis Fassbender
dfabender@uni-koblenz.de

Uni Koblenz

July 14, 2009

Introduction

Why do we need access control?

- We want to have one knowledge base that is accessed by all users, but...
- We want to prevent different users from retrieving different kinds of information
- Method to define who is allowed to access what information is needed

Introduction

Why do we need access control?

- We want to have one knowledge base that is accessed by all users, but...
- We want to prevent different users from retrieving different kinds of information
- Method to define who is allowed to access what information is needed

Goals of this presentation

- Create Description Logic ontology representing a simple social network
- Use views to control access and enforce privacy policy

Access Control on the Semantic Web

Questions

- What are views?
- What is an ontology?
- What is Description Logic?
- How can views be used to restrict access to Description Logic ontologies?

Access Control on the Semantic Web

Questions

- What are views?
- What is an ontology?
- What is Description Logic?
- How can views be used to restrict access to Description Logic ontologies?

Views

- Traditionally used in relational database world
- Filter data that user can access
- Can combine data from different relations
- Can be used to hide sensitive data from user

Views - Example

users				
id	**nick**	**password**	**real_name**	**address**
1	john	e64f6...	John Meyer	Austin, TX
2	kate	a2cd5...	Kate Stevenson	Miami, FL

users_public	
id	**nick**
1	john
2	kate

What is an ontology?

Ontologies

- Specify which concepts appear in a certain domain and how they are related to each other

What is an ontology?

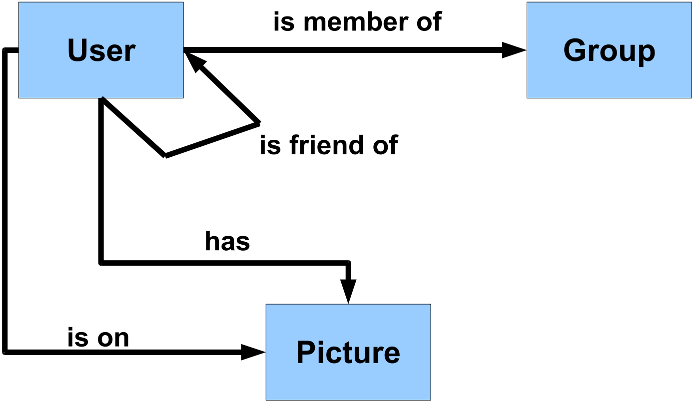
Ontologies

- Specify which concepts appear in a certain domain and how they are related to each other

Example: Simple social network

- Concepts: User, Group, Picture
- Relations:
  - Users are friends with other users
  - Users are members of groups
  - Users have pictures
  - Users are shown on pictures

Example: Ontology representing social network



Introduction to Description Logic

- DL knowledge base consists of TBox and ABox
- Knowledge represented by concepts and roles

Introduction to Description Logic

- DL knowledge base consists of TBox and ABox
- Knowledge represented by concepts and roles

ABox

User(*john*)
User(*kate*)
Group(*students_in_koblenz*)
IsFriendOf(*john*, *kate*)
IsMemberOf(*john*, *students_in_koblenz*)

Introduction to Description Logic

- DL knowledge base consists of TBox and ABox
- Knowledge represented by concepts and roles

ABox

$User(john)$
$User(kate)$
$Group(students\_in\_koblenz)$
$IsFriendOf(john, kate)$
$IsMemberOf(john, students\_in\_koblenz)$

TBox

$User \sqsubseteq \neg Group$

Introduction to Description Logic - Interpretations

Interpretations

- Defined as $I = (\triangle^I, \cdot^I)$
- Assumption: objects in $\triangle^I$ = individuals in ABox
- Assign sets of objects to concepts
- Assign sets of tuples of objects to roles

Introduction to Description Logic - Interpretations

ABox

$User(john)$
$User(kate)$
$Group(students\_in\_koblenz)$
$IsFriendOf(john, kate)$
$IsMemberOf(john, students\_in\_koblenz)$

TBox

$User \sqsubseteq \neg Group$

Introduction to Description Logic - Interpretations

ABox

$User(john)$
$User(kate)$
$Group(students\_in\_koblenz)$
$IsFriendOf(john, kate)$
$IsMemberOf(john, students\_in\_koblenz)$

TBox

$User \sqsubseteq \neg Group$

Possible interpretation

$\triangle^I = \{john, kate, students\_in\_koblenz\}$
$User^I = \{john, kate\}$
$Group^I = \{students\_in\_koblenz\}$
$IsFriendOf^I = \{(john, kate)\}$
$IsMemberOf^I = \{(john, students\_in\_koblenz)\}$

Queries in Description Logic

- Queries represented by Horn clauses
- Predicates represent concepts and roles

Queries in Description Logic

- Queries represented by Horn clauses
- Predicates represent concepts and roles

Example

Query: $q(x) \leftarrow \mathit{IsFriendOf}(\mathit{john}, x)$

Queries in Description Logic

- Queries represented by Horn clauses
- Predicates represent concepts and roles

Example

Query: $q(x) \leftarrow \mathit{IsFriendOf}(\mathit{john}, x)$
Remember: $\mathit{IsFriendOf}^I = \{(\mathit{john}, \mathit{kate})\}$

Queries in Description Logic

- Queries represented by Horn clauses
- Predicates represent concepts and roles

Example

Query: $q(x) \leftarrow \mathit{IsFriendOf}(\mathit{john}, x)$
Remember: $\mathit{IsFriendOf}^I = \{(\mathit{john}, \mathit{kate})\}$
Answer: $q^I = \{(\mathit{kate})\}$

Queries in Description Logic

- Queries represented by Horn clauses
- Predicates represent concepts and roles

Example

Query: $q(x) \leftarrow \mathit{IsFriendOf}(\mathit{john}, x)$
Remember: $\mathit{IsFriendOf}^I = \{(\mathit{john}, \mathit{kate})\}$
Answer: $q^I = \{(\mathit{kate})\}$

- Certain answer to q w.r.t. $W = \{I_1, ..., I_n\}$: Tuple that is in every q^{I_i}
- Shorthand notation for set of certain answers: $\mathit{cert}(q, W)$

Queries in Description Logic

- Queries represented by Horn clauses
- Predicates represent concepts and roles

Example

Query: $q(x) \leftarrow IsFriendOf(john, x)$
Remember: $IsFriendOf^I = \{(john, kate)\}$
Answer: $q^I = \{(kate)\}$

- Certain answer to q w.r.t. $W = \{I_1, ..., I_n\}$: Tuple that is in every q^{I_i}
- Shorthand notation for set of certain answers: $cert(q, W)$
- Views are represented by queries
- $v(x) \leftarrow IsFriendOf(john, x)$ is view that allows user to retrieve all of john's friends

View-based query answering

- Given: DL knowledge base K
- $MOD(K)$ = set of all models of K
- T = TBox of K
- $V = \langle v_1, ..., v_n \rangle$: views, e.g. $v_1(x) \leftarrow IsFriendOf(john, x)$
- $E = \langle e_1, ..., e_n \rangle$: certain extension of V
- $e_i = cert(v_i, MOD(K))$, e.g. $e_1 = \{(kate)\}$

View-based query answering

- Given: DL knowledge base K
- $MOD(K)$ = set of all models of K
- T = TBox of K
- $V = \langle v_1, \ldots, v_n \rangle$: views, e.g. $v_1(x) \leftarrow IsFriendOf(john, x)$
- $E = \langle e_1, \ldots, e_n \rangle$: certain extension of V
- $e_i = cert(v_i, MOD(K))$, e.g. $e_1 = \{(kate)\}$

Solution

Set W of interpretations is solution to (T, V, E) if:

- All interpretations in W are models for T (not necessarily for K!)
- $cert(v_i, W) = e_i$

Tuple a is valid view-based answer to q w.r.t. (T, V, E) if $a \in cert(q, W)$ for all solutions W

Practical Example: A Simple Social Network

Concepts

- User
- Group
- Picture

Roles

- IsFriendOf
- IsMemberOf
- HasPicture
- IsOnPicture

We'll refer to our complete knowledge base (TBox T + ABox A) as K

Practical Example: A Simple Social Network

TBox

$Group \sqsubseteq \neg Picture$

$User \sqsubseteq \neg Picture$

$User \sqsubseteq \neg Group$

- Group, User, and Picture are disjoint

Practical Example: A Simple Social Network

ABox

$User(john)$
$User(steve)$
$User(kate)$

$Group(students\_in\_koblenz)$

$Picture(pic1\_jpg)$
$Picture(pic2\_jpg)$

$IsMemberOf(john, students\_in\_koblenz)$
$IsMemberOf(kate, students\_in\_koblenz)$

$HasPicture(john, pic1\_jpg)$
$HasPicture(john, pic2\_jpg)$

$IsFriendOf(john, kate)$

$IsOnPicture(steve, pic1\_jpg)$

Practical Example: A Simple Social Network

Privacy policy

- Users cannot see members of a group they're not a member of:

$$v_1(x, y) \leftarrow \mathit{User}(\$id), \mathit{IsMemberOf}(\$id, x), \mathit{IsMemberOf}(y, x)$$

Practical Example: A Simple Social Network

Privacy policy

- Users cannot see members of a group they're not a member of:

  $$v_1(x, y) \leftarrow User(\$id), IsMemberOf(\$id, x), IsMemberOf(y, x)$$

- Users can only see pictures of friends and pictures they are on:

  $$v_2(x, y) \leftarrow User(\$id), HasPicture(x, y), IsFriendOf(\$id, x)$$
  $$v_2(x, y) \leftarrow User(\$id), HasPicture(x, y), IsOnPicture(\$id, y)$$

Practical Example: A Simple Social Network

Privacy policy

- Users cannot see members of a group they're not a member of:

$$v_1(x, y) \leftarrow User(\$id), IsMemberOf(\$id, x), IsMemberOf(y, x)$$

- Users can only see pictures of friends and pictures they are on:

$$v_2(x, y) \leftarrow User(\$id), HasPicture(x, y), IsFriendOf(\$id, x)$$
$$v_2(x, y) \leftarrow User(\$id), HasPicture(x, y), IsOnPicture(\$id, y)$$

Instantiated views for steve

$$v_1(x, y) \leftarrow User(steve), IsMemberOf(steve, x), IsMemberOf(y, x)$$
$$v_2(x, y) \leftarrow User(steve), HasPicture(x, y), IsFriendOf(steve, x)$$
$$v_2(x, y) \leftarrow User(steve), HasPicture(x, y), IsOnPicture(steve, y)$$

A useful interpretation

Consider the following interpretation I:

$\triangle^I =$
$\{john, steve, kate, students\_in\_koblenz, pic1\_jpg, pic2\_jpg\}$
$User^I = \{john, steve, kate\}$
$Group^I = \{students\_in\_koblenz\}$
$Picture^I = \{pic1\_jpg, pic2\_jpg\}$
$IsMemberOf^I =$
$\{(john, students\_in\_koblenz), (kate, students\_in\_koblenz)\}$
$HasPicture^I = \{(john, pic1\_jpg), (john, pic2\_jpg)\}$
$IsFriendOf^I = \{(john, kate)\}$
$IsOnPicture^I = \{(steve, pic1\_jpg)\}$

- I is a model for K
- Every fact in I is in every other model for K as well
- $cert(v_i, \{I\}) = cert(v_i, MOD(K))$

Practical Example: A Simple Social Network

Computing the certain extensions

Remember:

$v_1(x, y) \leftarrow User(steve), IsMemberOf(steve, x), IsMemberOf(y, x)$

Practical Example: A Simple Social Network

Computing the certain extensions

Remember:

$v_1(x, y) \leftarrow User(steve), IsMemberOf(steve, x), IsMemberOf(y, x)$

- $IsMemberOf(steve, x)$ not true in I for any x
  $\Longrightarrow$ body of v_1 always false under I (no answer)
  $\Longrightarrow$ $e_1 = cert(v_1, MOD(K)) = cert(v_1, \{I\}) = \{\}$

Practical Example: A Simple Social Network

Computing the certain extensions

Remember:

$v_2(x, y) \leftarrow User(steve), HasPicture(x, y), IsFriendOf(steve, x)$

$v_2(x, y) \leftarrow User(steve), HasPicture(x, y), IsOnPicture(steve, y)$

Practical Example: A Simple Social Network

Computing the certain extensions

Remember:

$v_2(x, y) \leftarrow User(steve), HasPicture(x, y), IsFriendOf(steve, x)$
$v_2(x, y) \leftarrow User(steve), HasPicture(x, y), IsOnPicture(steve, y)$

- $IsFriendOf(steve, x)$ not true in I for any x
  $\implies$ first body of v_2 always false under I (no answer)
- But: $User(steve)$, $HasPicture(john, pic1\_jpg)$, $IsOnPicture(steve, pic1\_jpg)$ true in I
  $\implies$ $(john, pic1\_jpg)$ is answer under I
  $\implies$ $e_2 = cert(v_2, MOD(K)) = cert(v_2, \{I\}) = \{(john, pic1\_jpg)\}$

Practical Example: A Simple Social Network

Query entered by steve

$q_1(x) \leftarrow IsMemberOf(x, students\_in\_koblenz)$

Practical Example: A Simple Social Network

Query entered by steve

$q_1(x) \leftarrow IsMemberOf(x, students\_in\_koblenz)$

Remember:

$v_1(x, y) \leftarrow User(steve), IsMemberOf(steve, x), IsMemberOf(y, x)$
$v_2(x, y) \leftarrow User(steve), HasPicture(x, y), IsFriendOf(steve, x)$
$v_2(x, y) \leftarrow User(steve), HasPicture(x, y), IsOnPicture(steve, y)$

Practical Example: A Simple Social Network

Query entered by steve

$q_1(x) \leftarrow \mathit{IsMemberOf}(x, \mathit{students\_in\_koblenz})$

Remember:

$v_1(x, y) \leftarrow \mathit{User}(\mathit{steve}), \mathit{IsMemberOf}(\mathit{steve}, x), \mathit{IsMemberOf}(y, x)$
$v_2(x, y) \leftarrow \mathit{User}(\mathit{steve}), \mathit{HasPicture}(x, y), \mathit{IsFriendOf}(\mathit{steve}, x)$
$v_2(x, y) \leftarrow \mathit{User}(\mathit{steve}), \mathit{HasPicture}(x, y), \mathit{IsOnPicture}(\mathit{steve}, y)$

Note: $\{I\}$ is solution to (T, V, E)
Create I' where I' is identical to I except that $\mathit{IsMemberOf}^{I'} = \{\}$
$\implies \mathit{cert}(v_1, \{I'\}) = e_1, \mathit{cert}(v_2, \{I'\}) = e_2$
$\implies \{I'\}$ is solution, but $\mathit{cert}(q_1, \{I'\}) = \{\}$
$\implies q_1$ has no answer

Practical Example: A Simple Social Network

Query entered by steve

$q_2(x) \leftarrow HasPicture(john, x)$

Practical Example: A Simple Social Network

Query entered by steve

$q_2(x) \leftarrow HasPicture(john, x)$

Remember:

$v_2(x, y) \leftarrow User(steve), HasPicture(x, y), IsFriendOf(steve, x)$
$v_2(x, y) \leftarrow User(steve), HasPicture(x, y), IsOnPicture(steve, y)$

Practical Example: A Simple Social Network

Query entered by steve

$q_2(x) \leftarrow HasPicture(john, x)$

Remember:

$v_2(x, y) \leftarrow User(steve), HasPicture(x, y), IsFriendOf(steve, x)$
$v_2(x, y) \leftarrow User(steve), HasPicture(x, y), IsOnPicture(steve, y)$

$(john, pic1\_jpg) \in e_2$
$\Longrightarrow$ $(john, pic1\_jpg) \in cert(v_2, \{I\})$
$\Longrightarrow$ $HasPicture(john, pic1\_jpg)$ true in every solution
$\Longrightarrow$ $(pic1\_jpg)$ is answer to q_2. What about $(pic2\_jpg)$?

Practical Example: A Simple Social Network

Query entered by steve

$q_2(x) \leftarrow HasPicture(john, x)$

Remember:

$v_2(x, y) \leftarrow User(steve), HasPicture(x, y), IsFriendOf(steve, x)$
$v_2(x, y) \leftarrow User(steve), HasPicture(x, y), IsOnPicture(steve, y)$

$(john, pic1\_jpg) \in e_2$
$\implies (john, pic1\_jpg) \in cert(v_2, \{I\})$
$\implies HasPicture(john, pic1\_jpg)$ true in every solution
$\implies (pic1\_jpg)$ is answer to q_2. What about $(pic2\_jpg)$?

Construct I^* such that I^* is identical to I except that
$HasPicture^{I^*} = \{(john, pic1\_jpg)\}$ ($(john, pic2\_jpg)$ left out).
$cert(v_i, \{I^*\}) = cert(v_i, \{I\}) \implies \{I^*\}$ is solution
$(pic2\_jpg) \notin cert(q_2, \{I^*\}) \implies$ not an answer

Conclusion

What have we accomplished?

- Explained why access control in knowledge bases is necessary
- Shown how ontologies can be written in Description Logics
- Transferred concept of views from relational database world to logic domain
- Shown how views can be used to control access to DL ontologies

Conclusion

What have we accomplished?

- Explained why access control in knowledge bases is necessary
- Shown how ontologies can be written in Description Logics
- Transferred concept of views from relational database world to logic domain
- Shown how views can be used to control access to DL ontologies

Prospect for the future

- Increase in popularity of DL as ontology language?
- Complex DL ontologies with advanced access control mechanisms?