

Web data management and Webdamlog

Serge Abiteboul

INRIA Saclay & ENS Cachan



Context: social data

The focus is on “social” data versus “public” data

Heterogeneity

Terminology : different ontologies

Systems: personal machines, social network systems, etc.

Distribution: different localization

Security: different protocols

Quality: Incomplete information, inconsistencies

Consequence:

- Difficult to manage your own data
- Difficult to find information in your friends data
- Difficult to keep control over your own data

Where is the data?

Laptop

Information system at work

Smartphone

Car computer

Home computer

Tablets

Nplay Box

...

Mail systems

Contacts

Agenda

Svn on forge, google docs

Web sites, blogs, tweets

Facebook, linkedin

Picasa, Flickr, Youtube...

...

Also access to data of family, friends, companies, associations...

What kind of data?

Data: a picture, some music, a movie, a report, an email, etc.

Metadata: this picture was taken by Alice in ENS Cachan on ...

Ontos: Alice's ontology & mappings with her friends' ontologies

Localization: her pictures are on Picasa; her back-ups are at inria

Access control: her Facebook friends can see her pictures

Security: AliceCachan on Facebook; password is 123456

Annotations: Alice likes Elvis's website

Beliefs: Alice believes Elvis is alive

External knowledge: Bob keeps copies of Alice's pictures

Time, provenance, etc.

Motivating example

Alice : get me recent pictures of Bob in parties we were together!

What is going on:

- Find on Facebook who are Alice's friends
- For each answer, say Sue, find where Sue keeps her pictures
- Find the means to access Sue's pictures, perhaps via some friends

Issues: heterogeneity of distribution and access control/security

- Some keep their pictures on servers such as Picasa
- Some put them encrypted in a public DHT
- Some have them on smart phones with a particular social net app
- For some, she may have to prove she has the right to see them
- Etc.

Thesis: this is a distributed knowledge base

Examples

Data/metadata: `picture@alice-iPhone(34434.jpg,09/12/02009,...)`

Annotations: `tag@delicious.com("wikipedia.org", dictionary)`

Localization: `where@alice(pictures, Picasa/AliceCachan)`

Access rights: `right@picasa/smith(pictures,friends,read)`

...

Information as logical statements

Each information belongs to a principal

A **physical principal**: alice-laptop, alice-iPhone, picasa, facebook, dht-peer-124, ...

- Storage and processing capabilities
- A peer typically has a URL and can be sent query/update requests

[Relation-name@peer-name\(data,...,data\)](#)

A **virtual principal**: alice, bob, alice-friends

- A virtual principal relies on peers for storage and processing
- A virtual principal typically has a URI and some authentication means , e.g. RSA keys

Datalog massacred?

Webdamlog extends datalog

- Negation
- Updates
- **Distribution and time**

datalog is simple, beautiful, declarative

Webdamlog is

- not as simple,
- not as beautiful,
- more procedural



This is not datalog

Requirements

Distributed world

- Many peers
- Exchange data between them
- Search for information – an issue is to localize who has some data

Dynamic world

- Peers come and go
- One can discover new peers
- One may need to exchange knowledge/rules with new peers

Webdamlog

Facts

Facts are of the form **$m@p(a_1, \dots, a_n)$**

- m is a *relation* name
- p is a *peer* name
- a_1, \dots, a_n are data *values* (n is the arity of $m@p$)
- The set of data values includes the sets of relations and peer names

$\text{parent}@p$ in E

$\text{parent}@p(\text{"peter"}, \text{"paul"})$ extensional

$\text{ancestor}@p$ in I

$\text{ancestor}@p(\text{"adam"}, \text{"paul"})$ intentional

Rules

Rules are of the form

$\$R@\$P(\$U) \text{ :- (not) } \$R_1@\$P_1(\$U_1), \dots, \text{(not) } \$R_n@\$P_n(\$U_n)$

where

- $\$R, \R_i are relation terms
- $\$P, \P_i are peer terms
- $\$U, \U_i are tuples of terms

Safety condition

- $\$R, \P must appear positively bound in the body
- Each variable in a negative literal, must appear positively bound in body

Different kinds of rules

Consider rules at a peer loc

- Local rules: all its body predicates are from p

Local rules with extensional head

ext-s@loc(x,y) :- r@loc(x,y) % insert a fact in local database

ext-r@p(4,4) :- r@loc(3,3) % send a message to p

Local rules with local intentional head

int-t@loc(x,y) :- r@loc(x,y) % classical notion of idb as in datalog

Local rules with nonlocal intentional head

int-t@loc(x,y) :- r@loc(x,y) % **view delegation**

Nonlocal rules

int-t@loc(x,y) :- r@loc(x,y) % **general delegation**

Semantics of general delegation

(rule at p) $t@q(x,y) :- r@p(x,z), r@p'(z,y)$

Suppose that $r@p(1,2)$ holds, then p “installs” at p’ the rule

(rule at p’) $t@q(1,y) :- r@p'(2,y)$

Latter if $r@p(1,2)$ does not hold, then p “uninstalls” that rule

An alternative more databasish view

At p: $seed@p'(x,z) :- r@p(x,z)$ view delegation

At p': $t@q(x,y) :- seed@p'(x,z), r@p'(z,y)$ delegation

Example: Peer and relation reification

Peers and relations are data (reified)

Alice: get me the pictures where I am with Bob that are stored on friends smartphones?

result@alice(\$X, \$U, \$Meta) :-

 friends@facebook(alice,\$X),
 smartphone@Sndirectory(\$X,\$P),
 photos@\$P(\$U,\$Meta),
 contains@\$P(\$Meta, "Alice") , contains@\$P(\$Meta, "Bob")

Similar for relations

On-going works around Webdamlog

- Implementation
 - On top of the Bud datalog system of Berkeley
- Probabilistic Webdamlog
 - Imprecisions, contradictions
- Access control in a social network
 - Control the distribution of your data
 - Control the application that you install

