Webdam Exchange: A model for data exchange on the Web

Serge Abiteboul
INRIA Saclay & ENS Cachan

WTS, Nancy, 2010

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
SACLAY - ÎLE-DE-FRANCE

Webdam

Organization

Introduction

Representing *all* Web information as logical sentences

Specifying system policies using a datalog-style language

[Webdam system]

Conclusion

Introduction

Context: Web data management

- Scale (lots of users, servers, large volume of data)
- Incomplete information, inconsistencies (belief, trust)
- Terminology heterogeneity, ontologies (semantic Web)
- Distribution heterogeneity: social networks, P2P, DHT, gossiping…
- Security heterogeneity: login, https, crypto, hidden URL…
- The heterogeneity keeps increasing with new systems and new applications arriving

Consequence: difficulty to perform data integration/management

Consequence: impossibility to keep control over its own data

Context of the work presented here

ERC Grant **Webdam** on Web Data Management

Joint work with many colleagues & in particular Alban Galland's thesis

Thesis: Web data = distributed knowledge

Work plan

1. Represent *all* Web information as logical sentences
2. Specify system policies using a datalog-style language
3. Develop a system to validate these ideas

Motivation for the approach

- Facilitate the design/implementation of complex systems
- Facilitate the control/surveillance of complex systems
- Use reasoning to optimize query evaluation
- Use reasoning for semantics/ontologies
- Use reasoning to manage access control and protect data
- Use reasoning to analyze properties of systems

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.

Representing *all* Web information as logical sentences

The kind of information we are talking about

Data: e.g., pictures, movies, music, emails, ebooks, reports

Annotations: e.g., semantic tags in Picasa

Ontologies and multilingual: e.g., RDFS, OWL…

Localization: Bookmarks, knowledge such as Alice has an account in Facebook, Sue puts her pictures in Picasa

Access: e.g., login/password, access rights on servers, lists of friends

Services: e.g., search engines, yellow pages, dictionaries…

Time, provenance, trust, quality…

And more…

The underlying model

Two kinds of principals

System peer: alice-iPhone, Picasa, facebook, aliceLaptop…

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

Virtual principal: alice, aliceFriends, wtsCommunity

- A virtual principal rely on peers for storage and processing
- A virtual principal has an identity (URI)

Peers and virtual principals have information

- Personal: alice states bob is a friend friends@alice(bob)
- For others: facebook exports “friends@alice(bob)”
  exports@facebook(friends,alice,bob)

Logical statements: personal knowledge Relation@Principal(Data-tuple)

Data:	picture@alice-iPhone(34434.jpg,09/12/02009,…)
Annotations:	tag@delicious.com(“wikipedia.org”, dictionary)
Localization:	where@alice(pictures, Picasa/AliceSmith)
	where@alice(pictures, alice-iPhone)
Access data:	access@picasa/smith(“alice”, “HG-FT23”)
Access rights:	right@picasa/smith(pictures,friends,read)
	group@picasa/smith(friends,bob)
Services:	search@google.com(“WTS“,$X)
	addresse@pagesjaunes.fr(“John Doe”, Paris, $Y)

Etc.

Personal knowledge

Alice states Bob is a friend – friend@alice(bob)

Includes more information that is not shown here

Alice-iPhone states friend@alice(bob)

Some authentication information: signature

- Alice-iPhone who created the statement
- alice: the principal this statement is about

Time: The time the statement was created (local time on the iPhone)

The content (here bob) is possibly encrypted

- For whom it is encrypted, public key or date of the key

Logical statements: external knowledge exports@Peer(Relation,Peer,Data-tuple)

Knowledge about other principals with time and provenance

Some knowledge stored on Alice's laptop

Base facts: alicePC exports "friend@alice(bob)"

AC facts: alicePC exports "bob canRead myPictures@alice"

Localization alicePC exports "myPictures@alice storedAt sue"

Secrets alicePC exports readKey@bob

The logical statements include time and provenance information

External knowledge

alicePC exports“friend@alice(bob)” includes more information

Some authentication information: signature

- alicePC who created the statement

Time: The time the statement was created (local time on alicePC)

Provenance:

alicePC exports

« alice-iPhone exports “alice-iPhone states friend@alice(bob)” to alicePC »

to bob-iPhone

The content (friend@alice(bob)) possibly encrypted

The model covers a wide range of data

The model does not prescribe any particular distribution architecture

- Gossiping, DHT, centralized server
- Combination of these
- Based on an abstract notion of localization

The model does not prescribe any particular access control policy

- Documents in Web servers with access protected by login/password
- Documents protected by cryptographic keys in public sites
- Based on an abstract notion of secret and hint

Webdam Exchange

All this information forms a knowledge base

Each peer manages some portion of the knowledge base

Policies are implemented as rules

Specifying system policies using a datalog-style language

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
SACLAY – ÎLE-DE-FRANCE

Warning

This is very *on-going* work

In particular, syntax and semantics are not stable

I will simply try to illustrate what we are trying to achieve

Webdamlog

Convention: variables/terms start with $; constants with small letters

Facts are of the form m@p(a1,...,an) (sorted)

Rules are of the form

$R@$P($U) :- (not) $R1@$P1($U1), …, (not) $Rn@$Pn($Un)

where

- $R, $Ri are message terms
- $P, $Pi are peer terms
- $U, $Ui are tuples of terms
- Safety condition

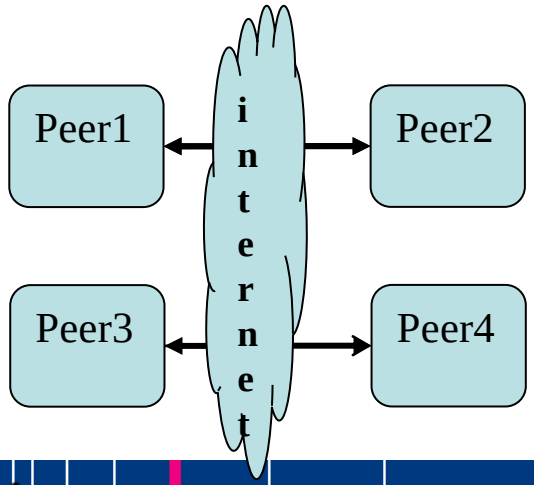
Intuition: if the body holds for some valuation v send the message v($R)@v($P)(v($U)) to the peer v($P)

Webdamlog

A finite set π of peers

Each peer p in π has a *program* **P(p)**, i.e. a finite set of *rules*

Each peer p in π has a *base* **I(p)**, consisting of a finite set of facts of the form m@p(u)



Semantics: in a state (P,I)

Choose randomly some p

- Evaluate P(p)(I(p))
- This defines directly the new state (P',I')(p) at p
- This defines the facts/rules that are added/removed to the state at each q ≠ p
- The changes to each q are installed *synchronously* – we will see how to avoid this if desired

Keep going (in a fair way)

Peer and message reification

Peers and messages as 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”)

Installing rules at another peer

Rule at Bob's iPhone to find Alice's data (ask systemL)

- $R@alice($X) :- true@systemL($R), $R@alice($X)

Rule at SystemL to find Alice's pictures (ask her iPhone)

- photo@alice($X) :- true@iPhoneAlice(), photo@alice($X)

Rule at Alice's iPhone to find Alice's pictures (look in local database)

- $R@alice($X) :- db3@iPhoneAlice(alice,$R,$X)

Rewriting of a rule at Bob's iPhone to get Alice's pictures

res@iPhoneBob($X) :- photo@alice($X) **query installed at bob**

res@iPhoneBob($X) :- true@systemL, photo@alice($X) **rule installed at systemL**

res@iPhoneBob($X) :- true@iPhoneAlice, photo@alice($X) **rule installed at iPhoneAlice**

res@iPhoneBob($X) :- db@iPhoneAlice(alice,photo,$X) **rule executed at iPhoneAlice**

Basic idea

Work in a dynamic world

One can discover new peers

One can interact with them after loading some rules

Managing rules at other peers

This is complex: instantiations of rules are installed at peers and later possibly removed

To handle that

1. Rules are installed
2. Their instantiations are controlled via “seed” relations
3. The seed can be seen as a view that is maintained

Security risk

- Someone else is installing code (rules) and data in a peer
- Need to be controlled

More refined asynchronicity

To model message from peer p to peer q, we use a “peer” net_{pq} that captures the network

Replace a call m@q(u) at p by $m@net_{pq}(u)$

Almost there

- net_{pq} just relays messages: $M@q($U) :- $M@$net_{pq}$($U)
- Problem: all messages from p to q in the net arrive at the same time

More realistic solution using time $m@net_{pq}(u,t)$

- t is the time of the send at p
- $M@q($U) :- $M@$net_{pq}$ ($U,$T), min ($T , $M@$net_{pq}$ ($U,$T))
- Using min aggregate function

Conclusion

Lots of works to do

Webdam Exchange model is now stabilizing

Webdamlog is still rapidly evolving

Implementation: slowly progressing

Many issues

- Concurrency: right revocation
- Optimization: link with the works on optimization in AXML
- Looking for a killer application

Verification of applications: not started yet

- Related to verification of system with data
- Verify what: access control is not violated, one gets all the information one has access to, diagnosis in case of violation

Implementation

Web: Persistent Webdam Exchange Peer

- All functionalities
- Database, cryptography, communication, wrapper for external systems (e.g. Facebook, DHT)

Web application for using the system

- Web application based on a transient WEP used as an avatar

WEPlight

- Developed for the iPad
- Limited functionalities but relies on a proxy
- Exports data on the device possibly encrypted

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
SACLAY – ÎLE-DE-FRANCE