Viewing the Web as a Distributed Knowledge Base

Serge Abiteboul
INRIA Saclay, Collège de France and ENS Cachan

Joint work with Emilien Antoine, Meghyn Bienvenu, Daniel Deutch, Alban Galland, Kristian Lyndbaek, Julia Stoyanovich, Jules Testard
The Web as a distributed knowledge base

- WebdamLog: a rule-based language for the Web
- The WebdamLog system
- Inconsistencies and uncertainty
- Conclusion
The Web

hypertext

universal library of text

and multimedia

personal/private data

social data
A typical Web user’s data

- **What kinds of data?** all kinds
  - *data*: photos, music, movies, reports, email
  - *metadata*: photo taken by Alice in Paris on ...
  - *ontologies*: Alice’s ontology and mapping with other ontologies
  - *localization*: Alice’s pictures are on Picasa, back-ups are at INRIA
  - *security*: Facebook credentials (Alice, 123456)
  - *annotations*: Alice likes Elvis’ website
  - *beliefs*: Alice believes Elvis is alive
  - *external knowledge*: Bob keeps copies of Alice’s pictures
  - *time, provenance*, ...
A typical Web user’s data

• What kinds of data? *all kinds*

• Where is the data? *everywhere*
  - laptop, desktop, smartphone, tablet, car computer
  - mail, address book, agenda
  - Facebook, LinkedIn, Picasa, YouTube, Tweeter
  - svn, Google docs
  - also access to data / information of family, friends, companies associations
A typical Web user’s data

• What kinds of data? *all kinds*
• Where is the data? *everywhere*
• What kind of organization? *heterogeneous*
  — terminology: different ontologies
  — systems: personal machines, social networks
  — distribution: different localization
  — security: different protocols
  — quality: incomplete / inconsistent information
Example of processing

Alice and Bob are getting engaged. Their friends want to offer them an album of photos where they are together.

To make such a photo album:

- Find friends of Alice & Bob (say with Facebook)
- for each friend, find where she keeps her photos (say, Picassa)
  - find the means to access her photos possibly via friends
  - find the photos that feature Bob and Alice together, e.g., using tags or face recognition software
- possibly ask someone to verify the results

Some reasoning is needed to execute these tasks automatically!
A typical Web user

- Overwhelmed by the mass of information
- Cannot find the information needed
- Is not aware of important events
- Cannot manage/control how others access and use his/her own data
How can systems help?

• We need to move from a Web of text to a Web of knowledge
  — In the spirit of semantic Web

• To better support user needs,
  — Systems need to analyze what is happening and construct knowledge
  — Systems should exchange knowledge
  — Systems should reason and infer knowledge
Thesis

All this forms a distributed knowledge base

with processing based on automated reasoning
Issues

• Distributed reasoning

• Exchanging facts and rules

• Contradictions

• Missing and noisy data

WebdamLog

Ignore for now
• The Web as a distributed knowledge base
  ▶ WebdamLog: a rule-based language for the Web
• The WebdamLog system
• Inconsistencies and uncertainty
• Conclusion
WebdamLog: a datalog-style language

Why datalog?

A prehistoric language by Web time...

- nice and compact syntax
- well-studied with many extensions
- recursion essential: cycles in the network

Not as simple/beautiful & more procedural

Needed for real Web applications!

WebdamLog is not datalog
WebdamLog: a datalog-style language

Extensional facts
friend("peter","paul") friend("paul","mary")
friend("mary","sue")

Datalog program
fof(x,y) :- friend(x,y)
fof(x,y) :- friend(x,z), fof(z,y)

Intentional facts
fof("peter","paul") fof("peter","mary") fof("peter","sue")
fof("paul","mary") fof("paul","sue")
WebdamLog

Extends datalog

• negation, updates, distribution, delegation, time

For a world that is

• distributed: autonomous and asynchronous peers
• dynamic: knowledge evolves; peers come and go

Influenced by

• Active XML (INRIA) - for distribution & intentional data
• Dedalus (UC Berkeley) - for time & implementation
Schema

\((\pi, E, I, \sigma)\)

\(\pi\) possibly infinite set of peer IDs

\(E\) set of extensional relations of the form \(m@p\)

\(I\) set of intentional relations of the form \(m@p\)

\(\sigma\) sorting function

for each \(m@p\), \(\sigma(m@p)\) is an integer (its sort)
Facts

Facts are of the form $m@p(a_1, ..., a_n)$, where

- $m$ is a relation name
- $p$ is a peer name
- $a_1, ..., a_n$ are data values ($n$ is the arity of $m@p$)

the set of data values includes the relations and peer names

Examples

- friend@my-iphone(“peter”, “paul”)  
  extensional
- fof@my-iphone(“adam”, “paul”)  
  intentional
Examples of facts


ontology: isA@yago.com("Elvis", theKing)

annotations: tags@delicious.com("wikipedia.org", encyclopedia)

localization: where@alice(pictures, picasa/alice)

access rights: right@picasa(pictures, friends, read)

security: secret@picasa/alice; public@picasa/alice
Rules

Rules are of the form

\[ R@P(U) : - (not) \ R_1@P_1(U_1), \ldots, (not) \ R_n@P_n(U_n) \]

where

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

Safety condition

- \( R \) and \( P \) must appear positively bound in the body
- each variable in a negative literal must appear positively bound in the body

Examples coming up, stay tuned

A term is a variable or a constant
A state \((I, \Gamma, \Gamma^*)\) : each peer \(p\) has

- extensional facts \(I(p)\), defining the local state of \(p\)
- local rules \(\Gamma(p)\), defining the program of \(p\)
- rules \(\Gamma^*(p,q)\) that have been delegated to \(p\) by some peer \(q\)
State transition

Choose some peer \( p \) randomly – asynchronously

Compute the transition of \( p \)

the database updates at \( p \)

the messages sent to other peers

the delegations of rules to other peers

Keep going forever

\[(I_0, \Gamma_0, \emptyset) \rightarrow (I_1, \Gamma_1, \Gamma_1^*) \rightarrow \ldots \rightarrow (I_n, \Gamma_n, \Gamma_n^*) \rightarrow \ldots\]

Fair sequence: each peer is selected infinitely often
The semantics of rules

Classification based on **locality** and **nature of head predicates** (intentional or extensional)

- Local rule at my-laptop: all predicates in the body of the rules are from my-laptop

| Local with local intentional head               | classic datalog          |
| Local with local extensional head              | database update          |
| Local with non-local extensional head          | messaging between peers  |
| Local with non-local intentional head          | view delegation          |
| Non-local                                      | general delegation       |
Local rules with local intentional head

Example: Rule at peer my-laptop

`friend` is extensional, `fof` is intentional

\[
\text{fof}@\text{my-iphone}(x, y) ::= \text{friend}@\text{my-iphone}(x, y)
\]

\[
\text{fof}@\text{my-iphone}(x, y) ::= \text{friend}@\text{my-iphone}(x, z), \text{fof}@\text{my-iphone}(z, y)
\]

`fof` is the transitive closure of `friend`

Datalog = WebdamLog with only local rules and local intentional head
Local rules with local extensional head

A new fact is inserted into the local database

\[
\text{believe@my-iphone}("Alice", \$\text{loc}) \leftarrow \\
\text{tell@my-iphone}(\$p,"Alice", \$\text{loc}), \\
\text{friend@my-iphone}(\$p)
\]
Local rules with non-local extensional head

A new fact is sent to an external peer via a message

\$message@$peer(\$name, “Happy birthday!”) :-

today@my-iphone(\$date),

birthday@my-iphone(\$name, \$message, \$peer, \$date)

Extensional facts:

today@my-iphone(March 6)

birthday@my-iphone("Manon", “sendmail”, “gmail.com”, March 6)

sendmail@gmail.com("Manon", “Happy birthday”)

Local rules with non-local intentional head

View delegation!

\[
\text{boyMeetsGirl}@\text{gossip-site}($girl, $boy) :- \\
\text{girls}@\text{my-iphone}($girl, $loc), \\
\text{boys}@\text{my-iphone}($boy, $loc)
\]

Semantics of \text{boyMeetsGirl}@\text{gossip-site} is a join of relations \text{girls} and \text{boys} from \text{my-iphone}

Formally, \text{my-iphone} delegates a rule \text{boyMeetsGirl}@\text{gossip-site}(g,b) for each \(g, b, l, \text{girls}@\text{my-iphone}(g,l), \text{boys}@\text{my-iphone}(b,l)\)
Non-local rules: general delegation

(at my-iphone): boyMeetsGirl@gossip-site($girl, $boy) :-
    girls@my-iphone($girl, $loc),
    boys@alice-iphone($boy, $loc)

Suppose that girls@my-iphone(“Alice”, “Julia's birthday”) holds.
Then my-iphone installs the following rule at alice-iphone
(at alice-iphone): boyMeetsGirl@gossip-site(“Alice”, $boy) :-
    boys@alice-iphone($boy, “Julia's birthday”)

When girls@my-iphone(“Alice”, “Julia's birthday”) no longer holds, my-iphone uninstalls the rule
Non-local rules: general delegation

(at my-iphone):  boyMeetsGirl@gossip-site($girl, $boy) :-
                    girls@my-iphone($girl, $loc),
                    boys@alice-iphone($boy, $loc)

An alternative, more database-ish, way of looking at this:

at my-iphone :  seed@alice-iphone($girl, $loc):-
                    girls@my-iphone($girl, $loc)  (delegation)

at alice-iphone :  boyMeetsGirl@gossip-site($girl, $boy) :-
                          seed@alice-iphone($girl, $loc),
                          boys@alice-iphone($boy, $loc)  (delegation)
Complexity of delegation: illustration

fof(x,y) :- friend(x,y)

(at p) fof@p(x,y) :- peers@p($q), friend@$q(x,y)

If peers@p contains 100 000 tuples

peers@p(q_1), ...., peers@p(q_{100,000})

This rule will install 100 000 rules!

for i=1 to 100 000 (at q_i) fof@p(x,y) :- friend@q_i(x,y)

Data complexity transformed into program complexity
Summary of results [PODS 2011]

• Formal definition of the semantics of WebdamLog
• Results on expressivity
  — the model with delegation is more general, unless all peers and programs are known in advance
• Convergence is very hard to achieve
  — positive WebdamLog
  — strongly stratified programs with negation
• The Web as a distributed knowledge base
• WebdamLog: a rule-based language for the Web
  • The WebdamLog system
• Inconsistencies and uncertainty
• Conclusion
WebdamLog peers

[demo ICDE 2011, WebDB 2011]

Support communication with other peers

Support common security protocols

Support wrappers to external systems such as Facebook

Manage knowledge

- store knowledge (facts and rules)
- exchange knowledge with other peers
- perform reasoning
WebdamLog peers

Web services

communication

security

engine
WebdamLog engine [ongoing work]

Based on Bud

- developed at UC Berkeley, implemented in Ruby, open-source
- supports Bloom - an extension of datalog
- implements communication between peers
- serious experiments
WebdamLog inference: beyond Bud

- Translation of WebdamLog to Bloom (Bud’s language)

- Features of WebdamLog not supported in Bud
  1. Variable relation and peer names
  2. Delegation: non-local rules, non-local relations in the body
  3. Adding and removing rules at runtime: needed because of delegation
Example of runtime inference

(rule₁ at p)  boyMeetsGirl@p($girl, $boy) :-
girls@p($girl, $loc),
boys@p($boy, $loc)

(rule₂ at q)  gossip@$peer($girl, $boy) :-
boyMeetsGirl@q($girl, $boy),
allPeers($peer)

(rule₃ at q)  boyMeetsGirl@p($girl, $boy) :-
gossip@p($girl, $boy)

direct knowledge

hearsay
Adding facts at runtime

Maintain a provenance graph for update management
Removing facts at runtime

Avoid recomputation at each update using provenance
Provenance graphs

- Records the history of derivation
- **Provenance semiring** semantics [Green et al. 07]
  - alternative or joint use of data
  - facts, rules, peers are nodes
- Useful for **performance optimization**
- Other uses
  - explain results to users
  - specify and verify **access rights**
• The Web as a distributed knowledge base
• WebdamLog: a rule-based language for the Web
• The WebdamLog system
  ▶ Inconsistencies and uncertainty
• Conclusion
Motivation

- **Contradictions** (in intentional or extensional data) come from
  - errors, lies, rumors, updates
  - FD violations: some think Alice was born in Paris, others that she was born in London
  - opinions: some think Brahms is great; others don’t

- **Uncertainty** comes from
  - lack of information
  - contradictions

- **Probabilities** may be used to measure uncertainty
  - 80% think Alice was born in Paris, 20% in London
  - sources: we observed that Peter is wrong 20% of the time
We consider reasoning in an uncertain and inconsistent world

We do this

• first for the centralized setting
• then with distribution
• finally with probabilities

Datalog + FDs
WebdamLog
and sampling
Datalog example

• Where is Alice?

• A relation

\[
\text{IsIn}(\text{person}, \text{city}, \text{peer})
\]

with the FD

\[
(\text{person}, \text{peer}) \rightarrow \text{city}
\]

peer believes person to be in city

• Consider a datalog rule

\[
\text{IsIn}(\$\text{per}, \$\text{city}, \$\text{p'}) :- \text{IsIn}(\$\text{per}, \text{city}, \$\text{p}), \text{friend}(\$\text{p'}, \$\text{p})
\]

\[
\text{IsIn}(<\text{Alice}, \text{London}, \text{Bob}>) \quad \text{IsIn}(<\text{Alice}, \text{Paris}, \text{Sue}>)
\]

\[
\text{friend}(\text{my-iphone}, \text{Bob}) \quad \text{friend}(\text{my-iphone}, \text{Sue})
\]
Datalog with nondeterministic fact-at-a-time semantics

**Immediate consequence operator:** a single fact is derived only if it does not contradict known facts

*A possible world* is a maximal consequence. Example:

\[
\text{IsIn}(\$\text{per}, \$\text{city}, \$p') \text{ :- IsIn}(\$\text{per}, \text{city}, \$p), \text{friend}(\$p', \$p) \\
\text{IsIn}(\text{Alice}, \text{London}, \text{Bob}) \text{ IsIn}(\text{Alice}, \text{Paris}, \text{Sue}) \\
\text{friend}(\text{my-iphone}, \text{Bob}) \text{ friend}(\text{my-iphone}, \text{Sue})
\]

Infer: \text{IsIn}(\text{Alice}, \text{Paris}, \text{my-iphone})

*In practice set-at-a-time semantics is more efficient*
Discussion

Inflationary non-deterministic semantic ("stubborn" choices)

Related to 2-stable models

Proof theory

• Possible facts NP-complete
• Sure facts coNP-complete

Many possible alternative semantics
Distributed setting: use WebdamLog

To simplify, we focus only on local and deductive rules

The semantics is inflationary and non-deterministic

A subtlety: Each peer has to recall the choices made to always make the same choice in the future (when talking to other peers): stubborn

The causes of uncertainty

• Uncertainty in base facts

• Uncertainty in the order of peer activations

• Uncertainty in choosing immediate consequences
Probabilities

Probabilistic interpretation to measure uncertainty

• For base facts, use independent probabilistic events
• Uniform distribution for the next peer to activate
• Uniform distribution in choosing the next immediate consequence
  ◁ Can be done efficiently if there is a single FD & more complicated otherwise
Example: captures voting

Bob’s rules

\[ \text{lsIn@p(x,y)} : \text{ Follower@bob(p), lsIn@bob(x,y)} \]
\[ \text{lsIn@bob(x,y)} : \text{ baselsIn@bob(x,y)} \]

Suppose each peer has similar rules

Claim: For acyclic networks, the probability of a peer inferring a fact is exactly its relative support at his friends

Note: this also give semantics for more complicated cases such as networks with cycles
Query answering

Resulting tuples of a query q have associated probabilities

Exact evaluation using c-tables

• Too costly in practice

Sampling technique

• Each peer makes probabilistic choices along the way

• Converges to the probability of q when the number of samples grows
• The Web as a distributed knowledge base
• WebdamLog: a rule-based language for the Web
• The WebdamLog system
• Inconsistencies and uncertainty

Conclusion
Thesis

Let us turn the Web into a distributed knowledge base with billions of users supported by billions of systems analyzing information extracting knowledge exchanging knowledge inferring knowledge
Contribution

WebdamLog

• A language for distributed data management [PODS 2011]
• Datalog with distribution, updates, messaging
• Main novelty: delegation

System implementation

• Handles heterogeneity, localization and access control [WebDB 2011]
• WebdamlExchange peer In Java [demo ICDE 2011]
• WebdamLog engine based on Bud – ongoing
Issues & Ongoing works

Query optimization

Probabilistic WebdamLog

• Explaining results to users: top-k proofs

Collaboration between peers to answer queries

Access control based on provenance

Verification of applications

Lots of fun & many open questions
Cambridge University Press, 2012

http://webdam.inria.fr/Jorge