Web data management

Serge Abiteboul
INRIA Saclay & ENS Cachan

Oxford, March 2010

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE
INRIA
centre de recherche SACLAY – ÎLE-DE-FRANCE

The opposite of Michael B.'s talk
- Real examples
- No theorem
- No complexity class

Context: Web data management

Scale: lots of servers, large volume of data

Servers are autonomous (heterogeneous also)

Data may be very dynamic, heavy update rates

Peers are possibly moving

Evolution:

Relation	→	Tree
Centralized	→	Distributed
Precise data	→	Incomplete, probabilistic
Precise schemas	→	Ontologies

In this talk: survey of works on the topic

Active xml:	2002-2008	EC Goostep & ANR project DocFlow
Webdam:	2008-2013	ERC

With many colleagues, in particular:

- Tova Milo (Tel Aviv) — Victor Vianu (UCSD)
- Luc Segoufin (INRIA) — Ioana Manolescu (INRIA)
- Georg Gottlob (Oxford) — Alkis Polyzotis (UCSC)
- Angela Bonifati (Cozenza) — Marie-Christine Rousset (Grenoble)

And PhD students

- Omar Benjelloun (Google) — Bogdan Marinoiu (SAP)
- Pierre Bourhis (INRIA) — Alban Galland (INRIA)
- Marco Manna (Roma) — Nicoleta Preda (Franhoffer)
- Zoe Abrams (Google) — Emmanuel Taropa (Google)
- Bogdan Cautis (Telecom Paris) — Spyros Zoupanos (INRIA)

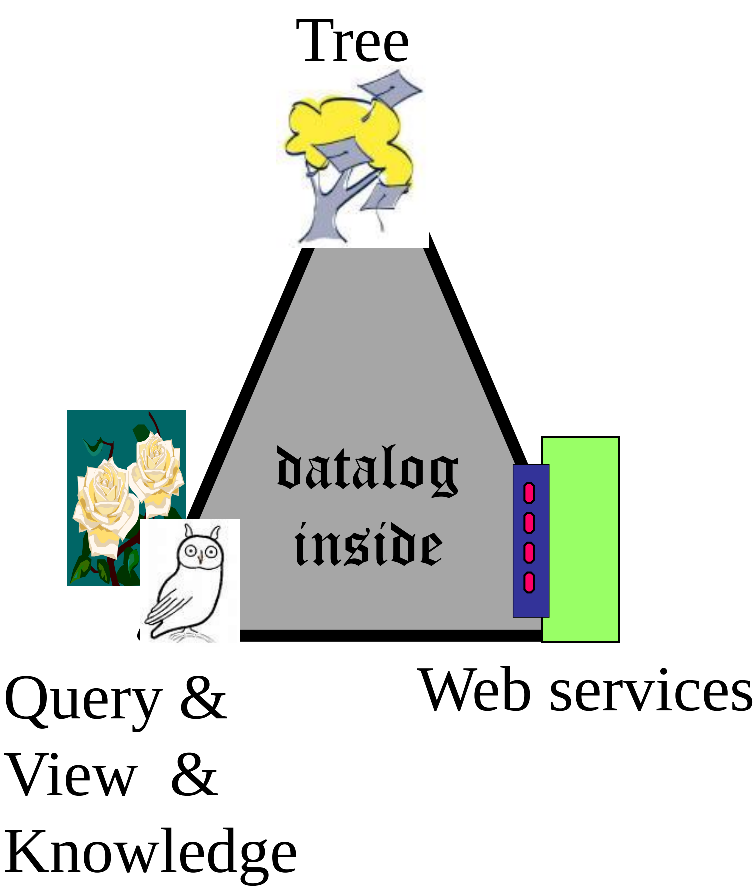
Key concepts

Data: Trees & xml

Queries: Xpath, Xquery

Processing: Web services

And datalog?



Organization

Introduction

Queries and views

- **The Active xml model**
- **Axml Algebra**
- **Distributed monitoring**

Sequencing and verification

- **Verification in Guarded Axml**
- **Axml Artifacts**
- **Workflow for active documents**

Access control

- **The Pastis model**

Conclusion

Buzwords

Web services, push, pull, streams, monitoring, interaction, communication

datalog 2.0

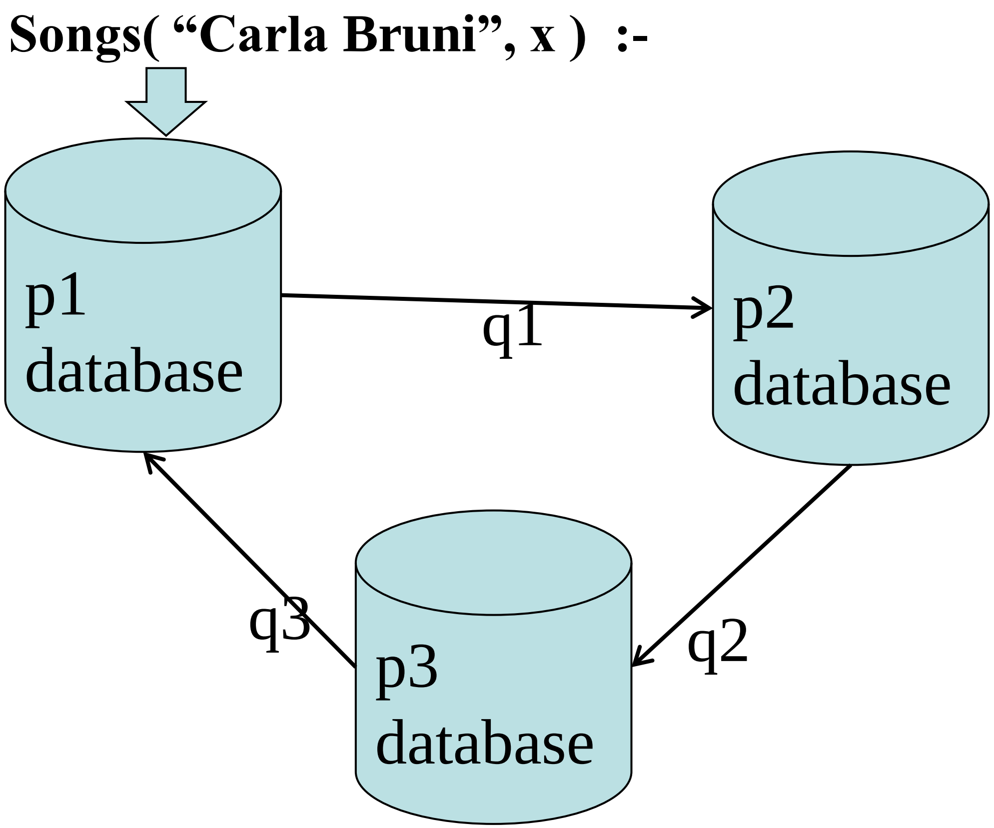
Verification, workflow

datalog 3.0

Knowledge, social networks, trust, beliefs

Model: Active xml

Example 1: Getting music over the net



Find me some songs of Carla Bruni, locally or somewhere

Of course, think of millions of peers with their own data

Active xml (see activexml.net)

Based on Web standards:

xml + Web services + Xpath/Xquery

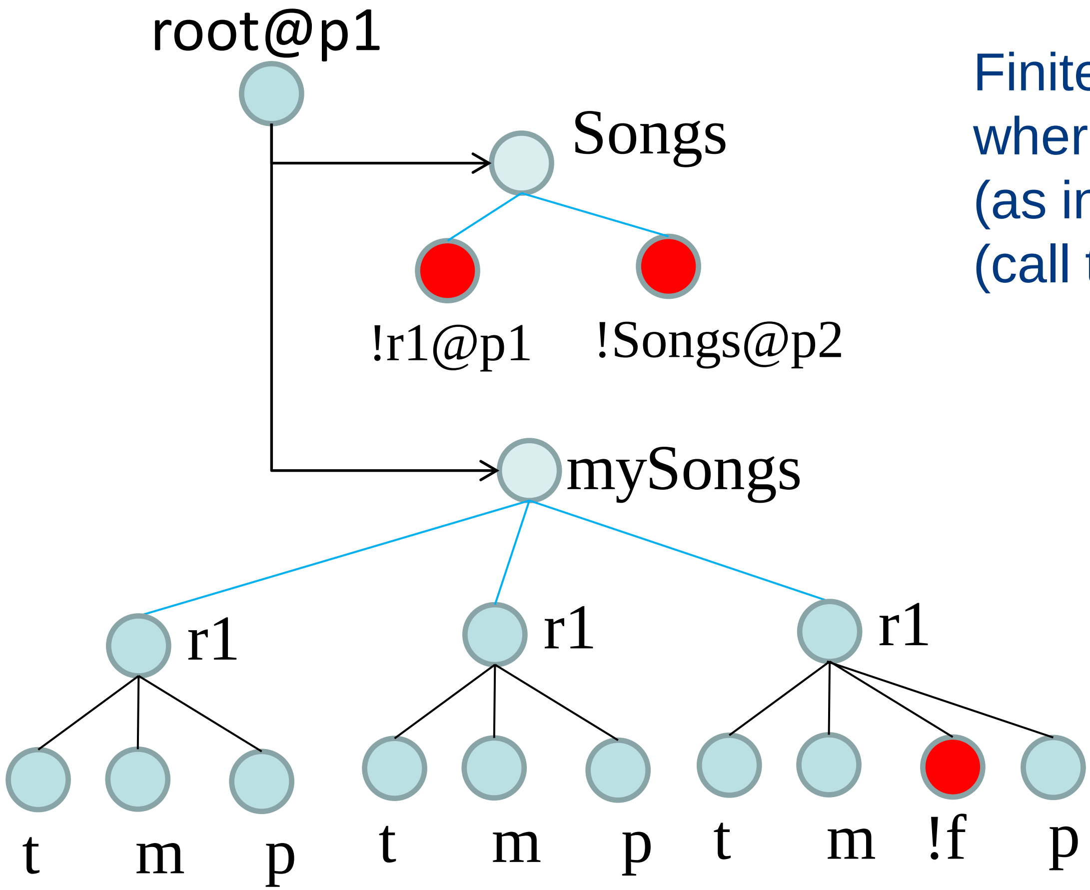
Simple idea

Exchange xml documents with embedded service calls

- Intentional data: get the data only when desired
- Dynamic data: If data sources change, the document changes
- Flexible data: adapt to the needs
- Function in push & pull mode; Sync and asynchronous

Embedding calls in data is an old idea in databases

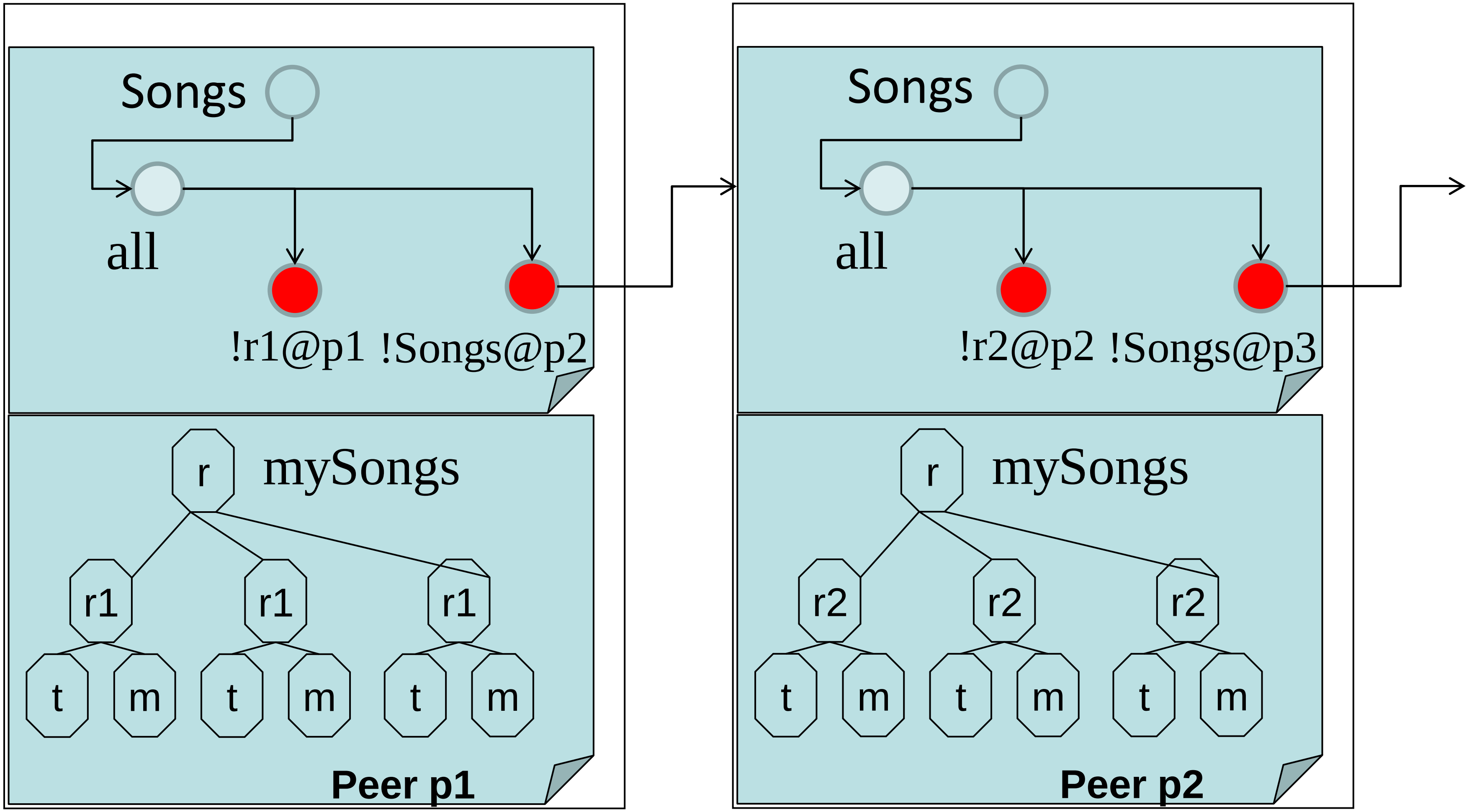
Axml



xml & Web services

Finite labeled unordered trees where labels are tags, data (as in xml) or function calls (call to Web services)

Activexml: xml documents with embedded service calls



This is distributed datalog over trees

Songs@p1(x,y) :- r1@p1(x,y)

Songs@p1(x,y) :- Songs@p2(x,y)

Songs@p2(x,y) :- r2@p2(x,y)

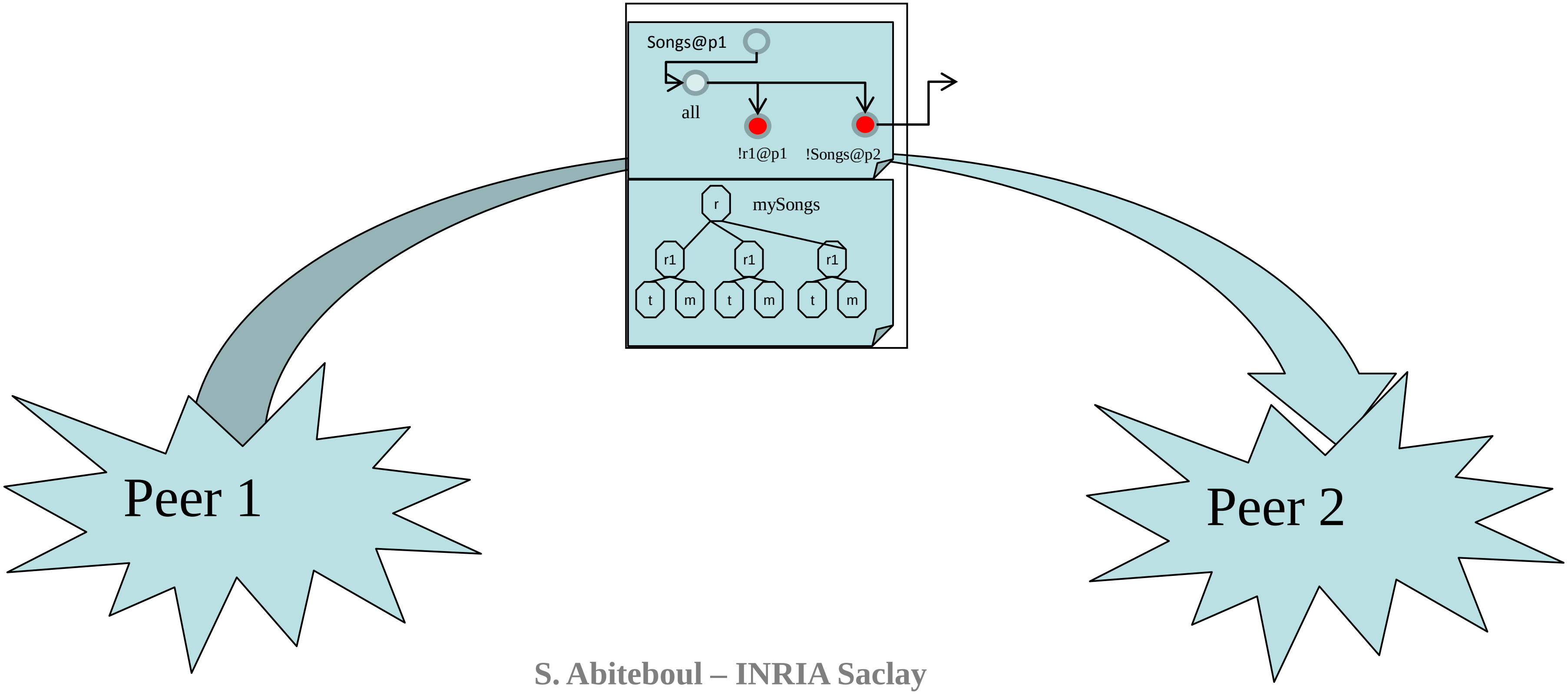
Songs@p2(x,y) :- Songs@p3(x,y)

Songs@p3(x,y) :- r3@p1(x,y)

Songs@p3(x,y) :- Songs@p1(x,y)

Songs@p1(“Carla Bruni”, x) :-

Moving data and logic around



The semantics of calls

When to activate the call?

- Explicit pull mode: active databases
- Implicit pull mode: deductive databases
- Push mode: query subscription

What to do with its result?

How long is the returned data valid?

What to send?

- *Phone number of the Prime Minister of France?*
- *Use whoswho.com then look in www.gouv.fr/phone*
- *Look for Fillon in www.gouv.fr/phone*
- *+33 1 56 00 00 07*

Active xml – cool idea – complex problems

Brings to a unique setting

distributed db,

deductive db,

active db,

stream data

warehousing & mediation

This is unreasonable? Yes!

Some works around Axml

The Axml system – open-source (on server, on smartphone)

The useful: Replication and query optimization

- How to evaluate a query efficiently by taking advantage of replication

The useful: Lazy query evaluation

- How to evaluate a query without calling all embedded services

The fun: Casting problem

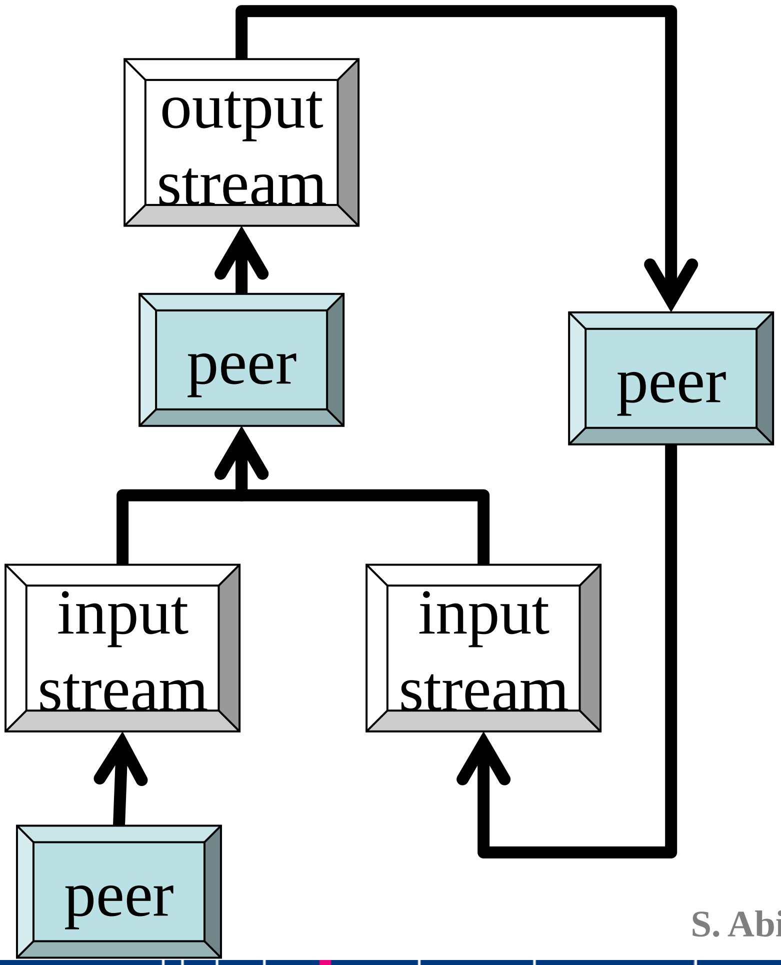
- Which functions to call to “match” a target type
- Active context-free games

The exotic: Diagnosis of communication systems

- The unfolding of the runs is described in Axml
- Datalog technology used for optimization

Query optimization & Axml Algebra

All is based on streams

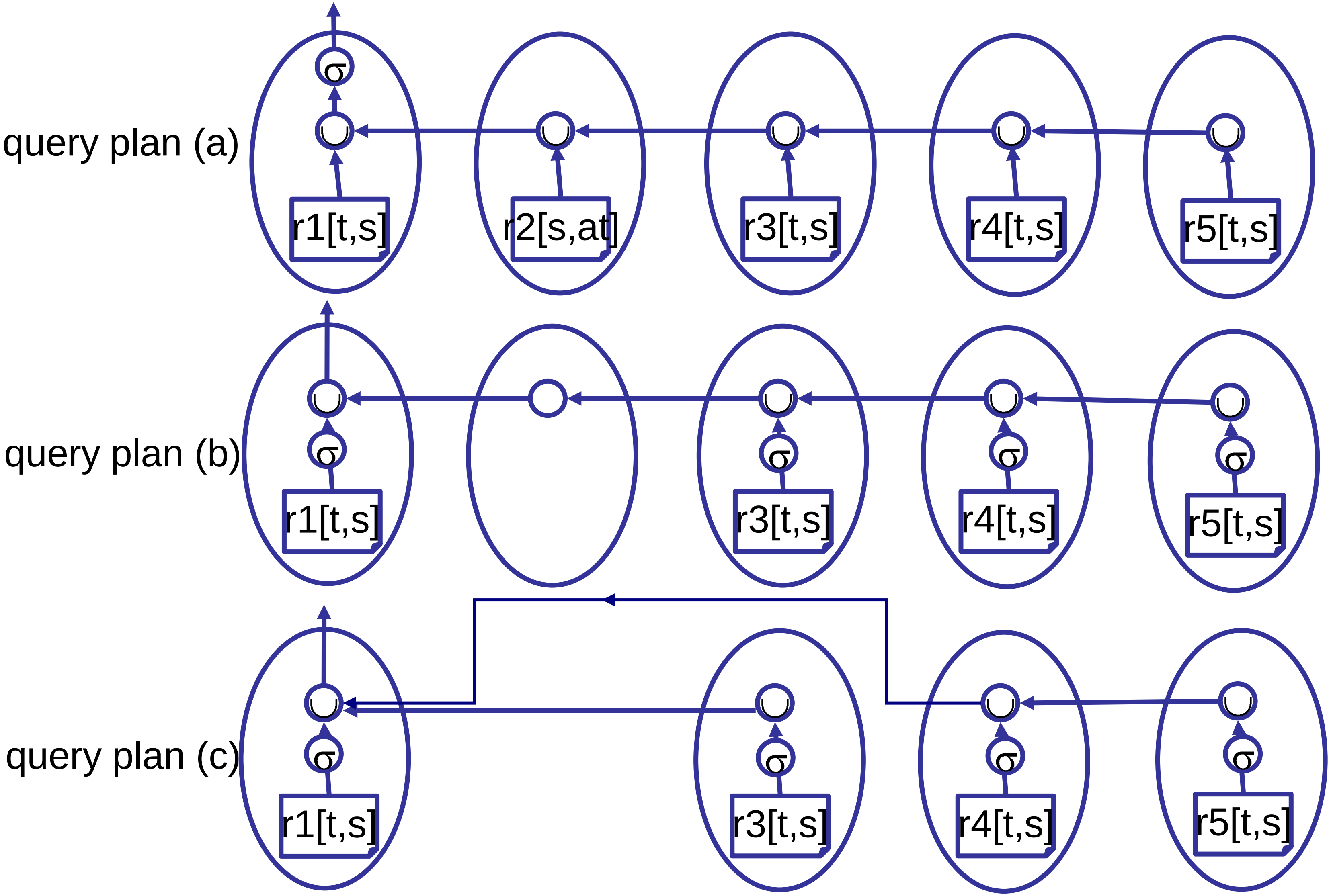


The local query processor knows how to optimize and compute stream queries

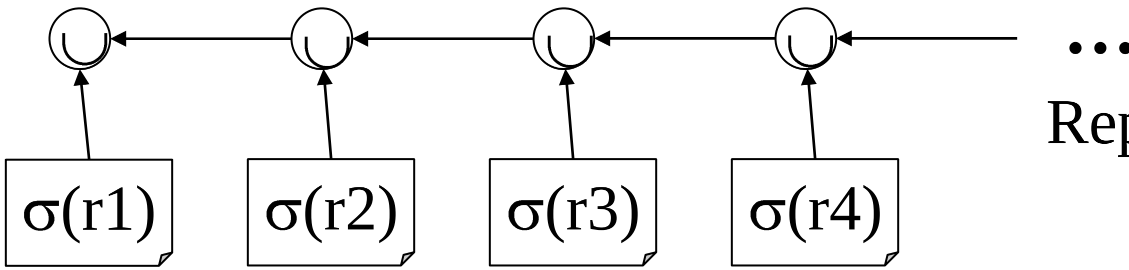
This is local – we don't care

Streams may be stored for future processing

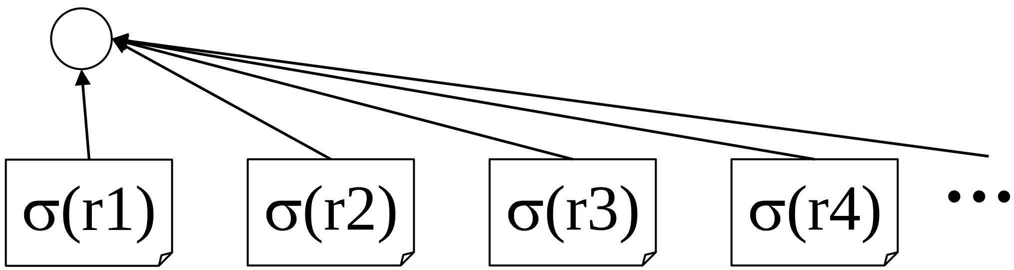
$\cup$ = outer join & $\sigma = \sigma_{s=\text{"Lhasa"}}$



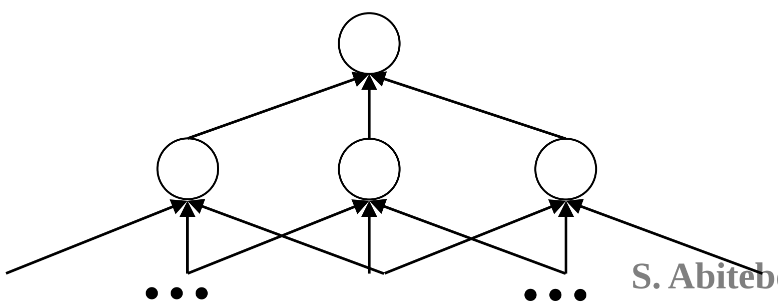
Optimization data/query routing



Repeated transfers

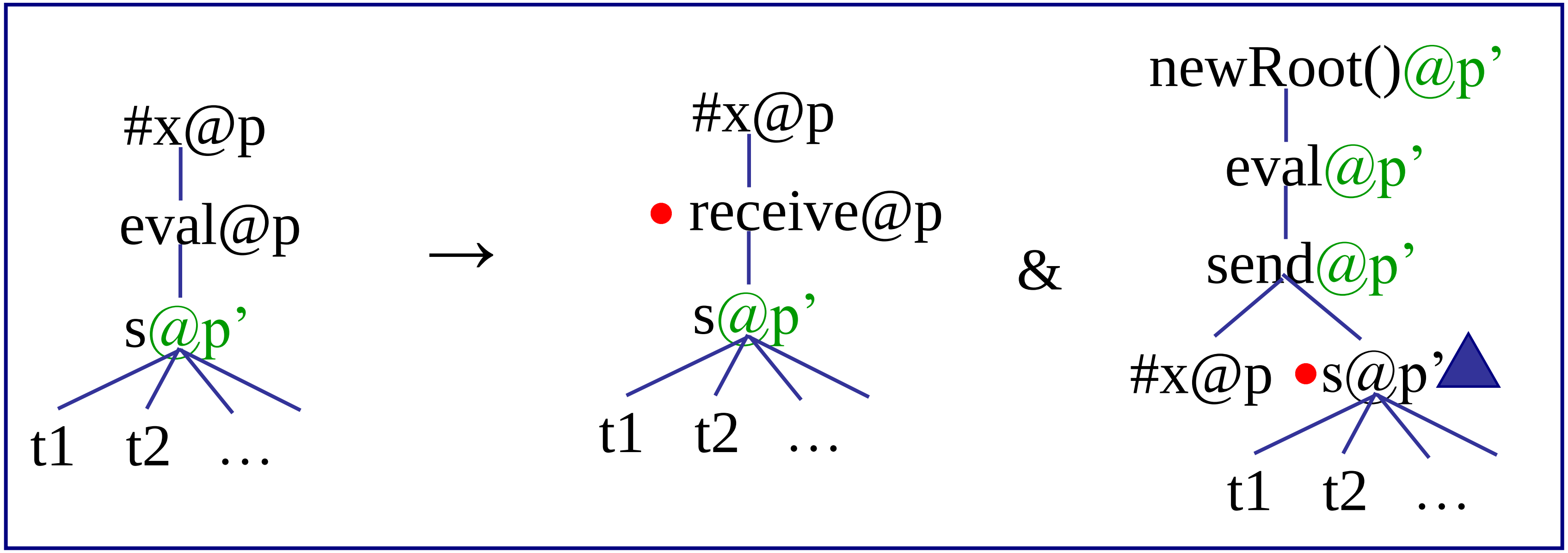


Data transfers reduced
More work for p1: merging all the streams



Hierarchical stream merging

Illustration of the algebraic rewrite rules



Site p asks p' to do the work and send the result to p

Monitoring

Monitoring distributed systems

Often distributed applications are very dynamic

- Content change rapidly
- Intense communications

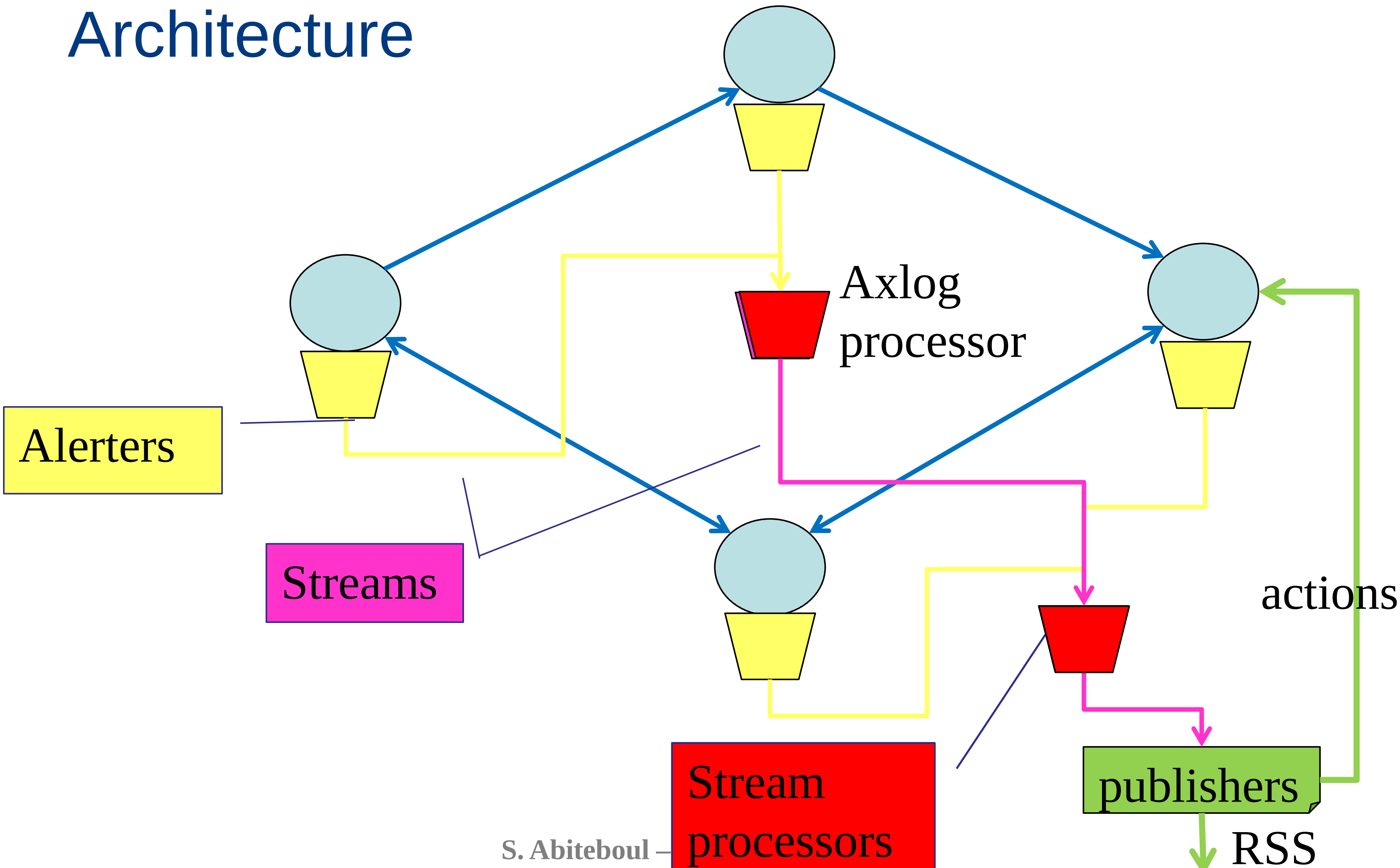
Complex and hard to control systems

- Many peers
- Peers are distributed
- Peers are autonomous
- Peers are sometimes unreliable and selfish
- Peers sometimes come and leave

Goal: **monitor such systems**

& support active features ala active databases

Architecture



Axlog principle = active document & query

Incoming streams of updates

The outgoing stream is defined by a query Q (tree-pattern + join)

Each time an incoming message arrives, it modifies the document so possibly the query result

The output stream specifies how to modify the view

Incremental view maintenance

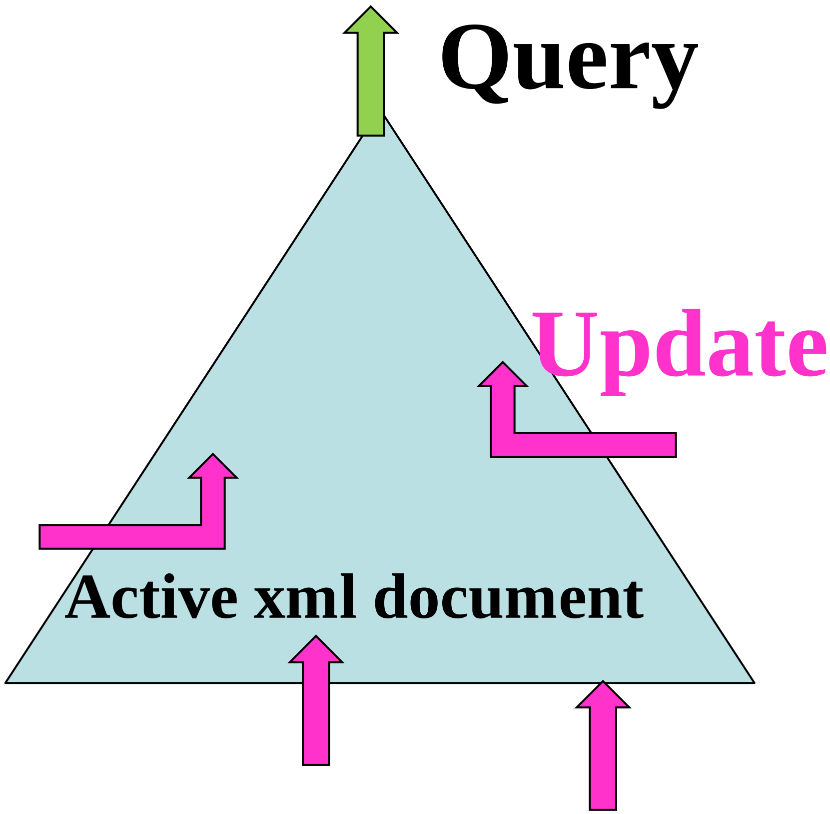
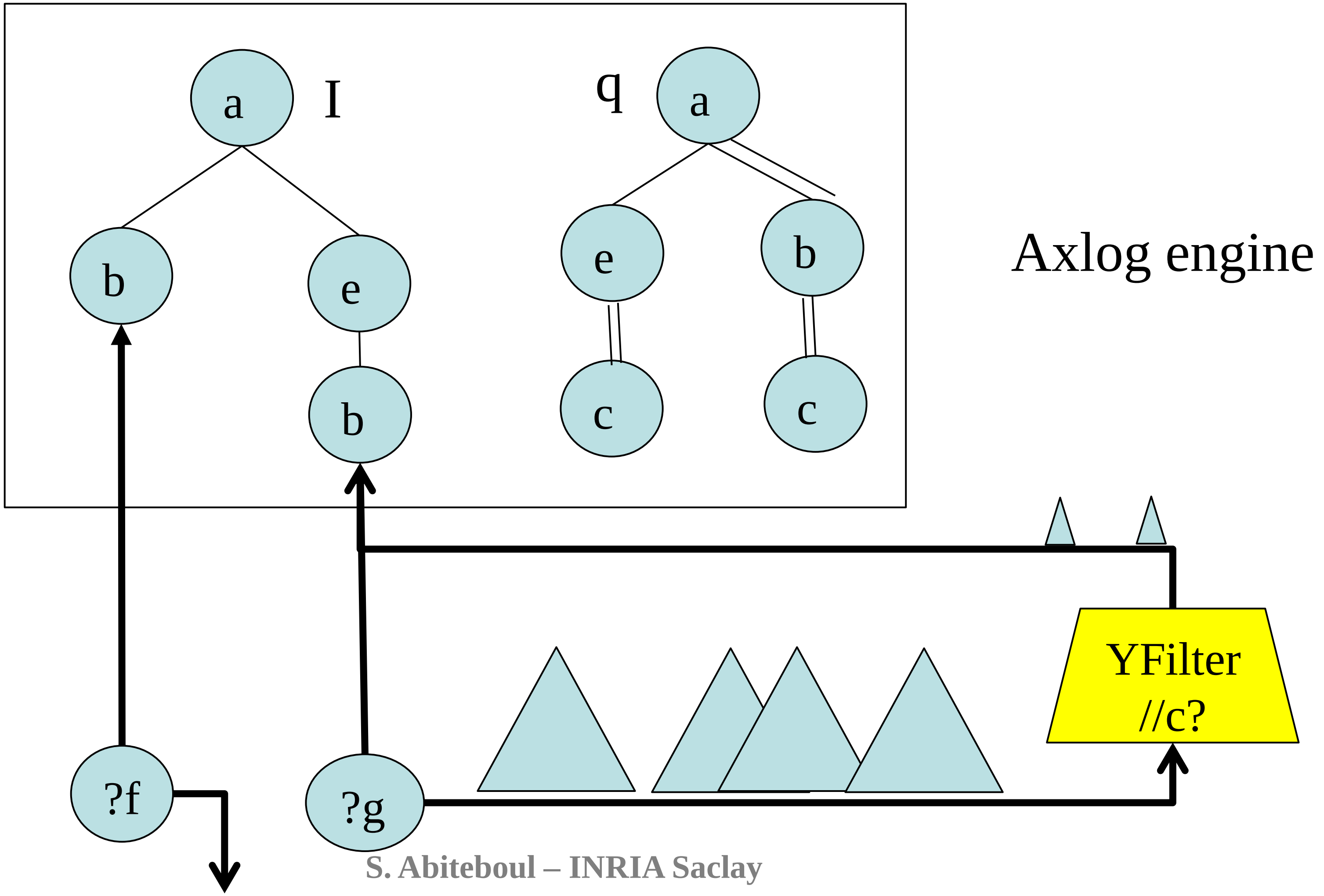


Illustration of optimization techniques: Filtering based on relevance



Axlog - continued

We have implemented an axlog engine

We use datalog to compute tree queries to benefit from

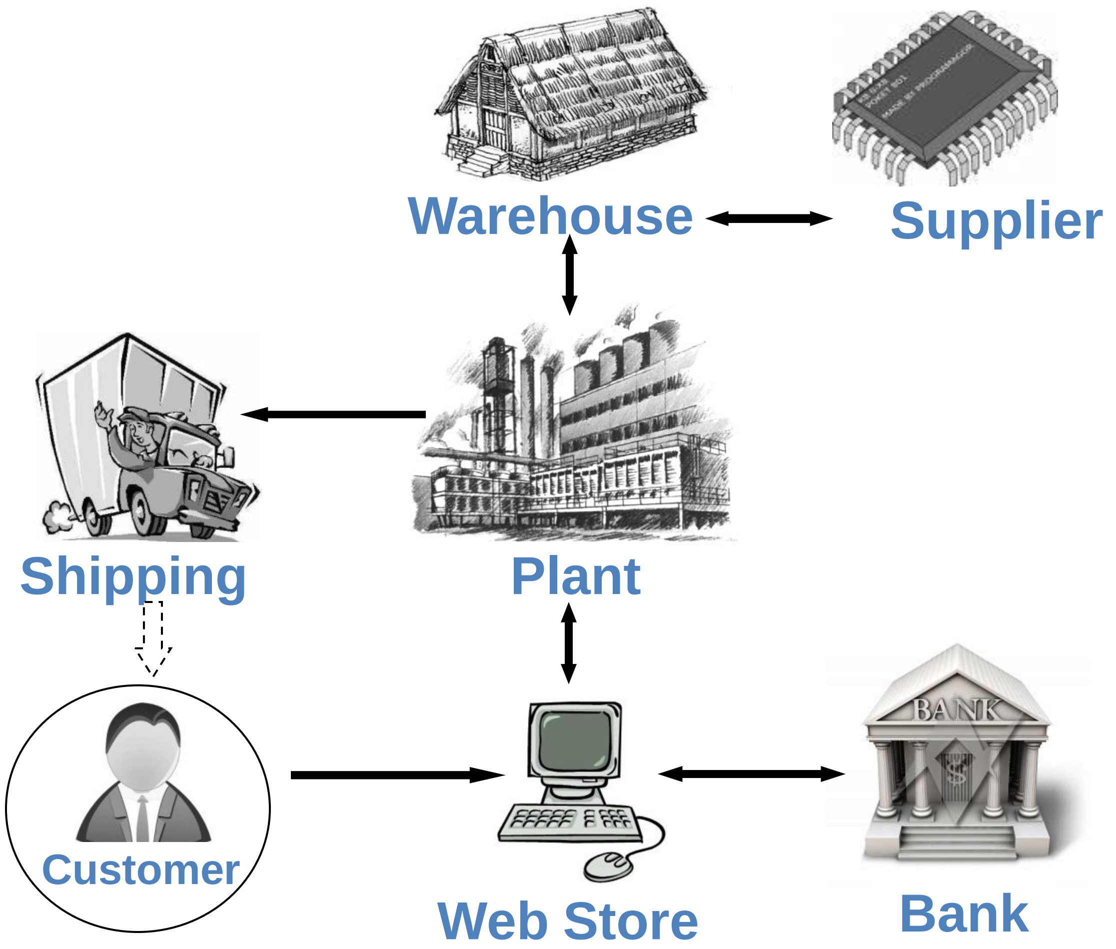
- Incremental view maintenance in datalog — Δ technique
- Query optimization in datalog — MagicSet
- Constraint query languages — CQL

We have developed specific techniques

- Compute (not incrementally) satisfiability and relevance
- Because of satisfiability more aggressive strategy that pure MagicSet
- Based on relevance, we filter the streams before they enter the datalog engine – very important savings – use of xml YFilter

Verification: Guarded Axml

Example 2: Dell Supply Chain



Issues

Verify the behavior of the system

Control the sequencing of the operations

A restricted model: guarded Axml

A restricted model so that verification can be performed

Based on imposing constraints on call activation/return: guards

Constraints on data: DTD + tree pattern formulas

Focus: deciding whether a service S satisfies a Tree-LTL sentence

- Decidable for bounded services***:*** no recursion
- Very high complexity – just a proof of feasibility
- Undecidable as soon as any of the syntactic restrictions are relaxed

Temporal formulas: Tree LTL

Boolean combinations of tree patterns & LTL operators

Syntax of Tree-LTL

φ :-pattern | φ and φ | φ or φ | not φ | **φ U φ | Xφ**

- pattern(X1,…,Xn) : all other variables are seen as existentially quantified
- **X: next U: until**
- Also **G: always? F: eventually. etc**

Tree-LTL sentence ∀φ

- All free variables are *quantified universally at the end*
- These are all the free variables from patterns

Example

Every webOrder is eventually completed (delivered or rejected)

$\forall$X [**G((T1(X) $\rightarrow$ F(T2(X) $\vee$ T3 (X)))]** where

T1(X): SYS [webOrder [Order-id [X]]]

T2(X): SYS [webOrder [Order-id [X] Delivered]]

T3(X) : SYS [webOrder [Order-id [X] Rejected]]

Active xml artifacts = Axart

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
SACLAY – ÎLE-DE-FRANCE

Artifact = Data & Control

Concept introduced by IBM Research

[Nigam & Caswell 03, Hull & Su 07]

Data-centric workflows

- A process is described by a document (possibly moving in the enterprise)
- The behavior of an artifact is specified by some constraints on how this document should evolve

Vs. state-transition-based workflows

- Based on some form of state transition diagrams (BPEL, Petri,…)
- Mostly ignore data

webOrder id=7787780
Customer
Name: John Doe
Address: Sèvres
Product: *committed*
Ref: PC 456
Factory: Milano
Parts: *waiting*
orderDate: 2009/07/24
Site: http:// d555.com
Payment: *done*
Bank-account …
Delivery: *not-active*

Axml Artifacts move on the Web

In webStore

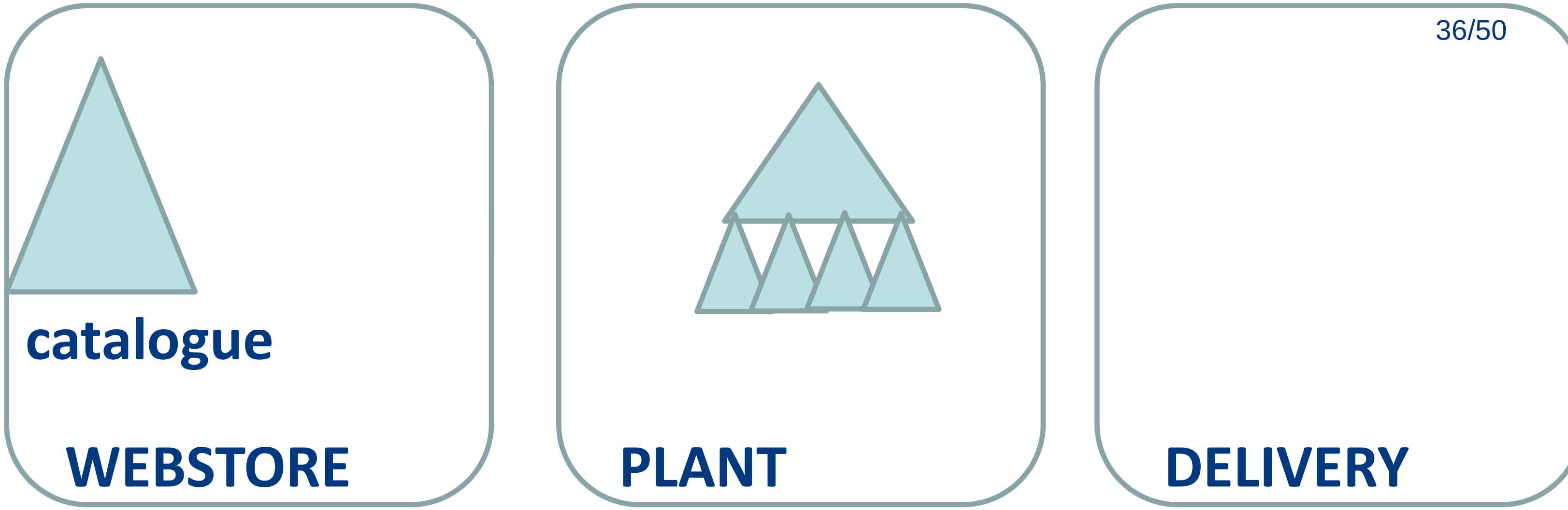
webOrder id=7787780
Customer
Name: John Doe
Address: Sèvres
Order selection: **on-going**
Ref: PC 456
Factory: *undecided*
Parts: *not-active*
orderDate: 2009/07/24
Site: http://d555.com
Payment: *pending*
Delivery: *not-active*

In plant

webOrder id=7787780
Customer
Name: John Doe
Address: Sèvres
Order selection : *committed*
Ref: PC 456
Factory: Milano
Parts: ***on-going***
orderDate: 2009/07/24
Site: http:// d555.com
Payment: *done*
Bank-account …
Delivery: *not-active*

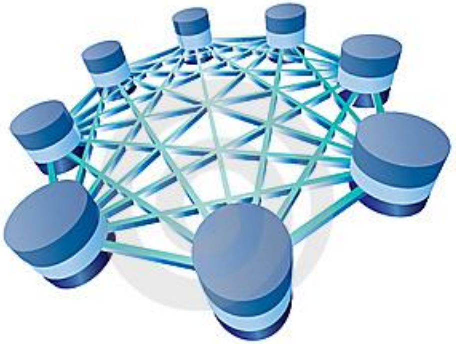
In delivery

webOrder id=7787780
Customer
Name: John Doe
Address: Sèvres
Order selection : *committed*
Ref: PC 456
Factory: Milano
Parts: *done*
orderDate: 2009/07/24
Site: http:// d555.com
Payment: *done*
Bank-account: CEIF-4457889
Delivery: ***on-going***
Address: Orsay



Axml
Artifact model

Sequencing of operations



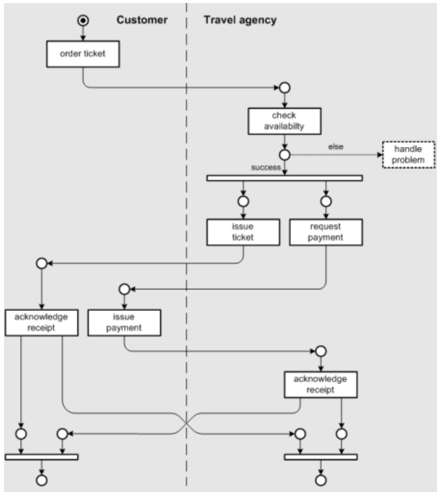
Different ways of expressing sequencing of tasks

- Guards: preconditions for function calls
- Transition-based diagrams
- Formulas in temporal logic

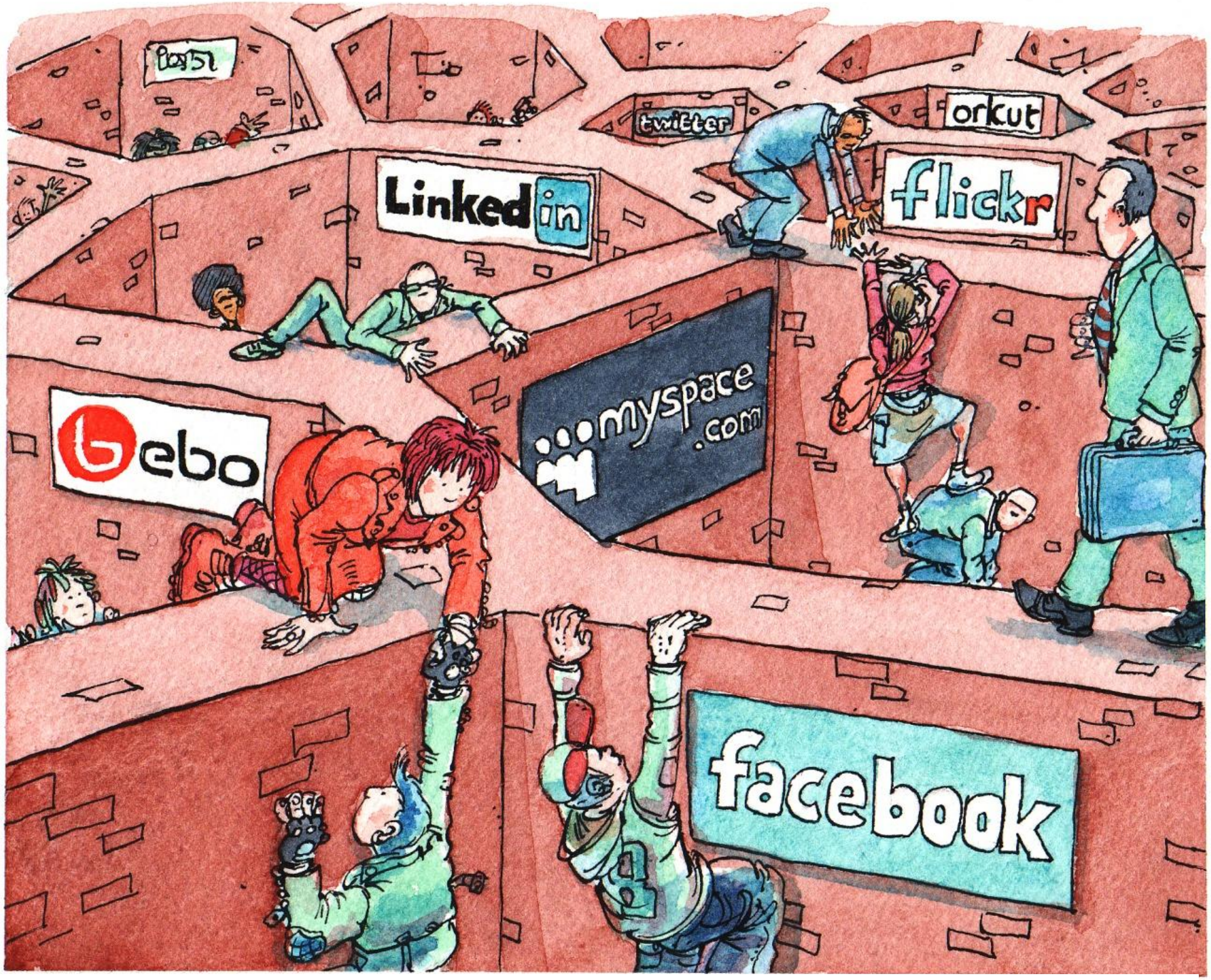
Study how they can simulate each other using some “scratch paper”

Data &

workflow



Access control



Example 3: Managing data in Social Networks

Issues

Control who can see your data

- The guy who is hiring you should not see the pictures of your last party

Have the right to be forgotten

- You should be allowed to remove these pictures entirely

Control who does what on your data

More and more concerns about that

This is all about access rights and querying/monitoring access controls and accesses

This is all about things we knew how to do in relational systems

Consider Alice's information data + knowledge

She has some data in

- In personal machines: laptop, smartphone
- At "trusted" SN Web sites: Facebook, LikedIn
- Replicated at friends: e.g., her last trip pictures at Bob
- In some not trusted DHT system
- In some trusted archiving system

She has keys for these systems (e.g., login/passwd)

She manages access rights to her data

She has some knowledge about where data is located

- Her data
- Her friend's data
- Other data

Of course she is lost
Any normal person would

Punch line

We treat all this information as a **distributed knowledge base** with

data (documents)

access control

keys

localization

time & provenance

The SomeWhere system François Goasdoué, Marie-Christine Rousset et al.

The Pastis model

The basis: principals

- Users (Alice), machines (Alice's system), systems (Facebook), groups (AliceFriends)

Access control is based on a distributed knowledge base

Base facts:	Alice states "Georg is Professor at Oxford"
External knowledge	Bob says 'Alice states "…"'
AC facts:	Alice states "Bob canRead myPictures@Alice"
Localization	Alice states "myPictures@Alice storedAt Bob
Keys	Alice states readKey@Bob

Accessing & updating information

Data

- Trees with references
- Collections (ala RSS feeds) represented as trees

Based on that one can locate and obtain information

Access rights

- Own – can also grant/revoke access rights
- Read
- Write
- Append/Remove from collections
- Corresponding cryptographic keys

Enforcing access control & auditing

Time and provenance are also recorded

All statements are authenticated (by the author and the access right needed for the statement)

Data is possibly encrypted so that it may be stored on untrested peers

Reasoning

In the knowledge base

- To locate data and answer queries – datalog again not surprisingly
- To optimize queries

About strategies/systems

- To check whether peer strategies are sound (no leak) and complete (no denial of data/update)

Combine that with SomeWhere: each peer has his own ontology + mapping between ontologies

Combine that with beliefs and trust: e.g.,
Alice believes Paul stores her pictures

Conclusion

Many other topics such as

Distributed xml design

- Work with Georg Gottloeb and Marco Manna
- General problem of constraint enforcement in distributed environment

Imprecise data

- Probabilistic xml with Pierre Senellart and Ievgeny Kharlamov

Concurrency control and transactions

Web data management

Lots of problems to investigate

Lots of challenges

We are still a long way from being able to teach properly Web data management

We are having lots of fun

Come and join us

And yes! Good old datalog plays an important role inside

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
SACLAY - ÎLE-DE-FRANCE

Modeling : Axart

[State] An artifact is an **object** with a universal **identity** (e.g., URI). Its state is **self-describing** (e.g., xml data) so that it may be easily transmitted or archived. It has a host (peer or artifact)

[Evolution] It is created, evolves in time (possibly space), hibernates, is reactivated or dies according to a declarative **logic** Its evolution is constrained by some laws, **workflow**

[Interface] An artifact interacts with the rest of the world via function calls, both **server** and **client**. It provides for communications, storage and processing for its **subartifacts**

[History] As in scientific workflows, an artifact has a **history** with **time** and **provenance** that may be recorded and queried

Artifact = business object & process/task & actor/service