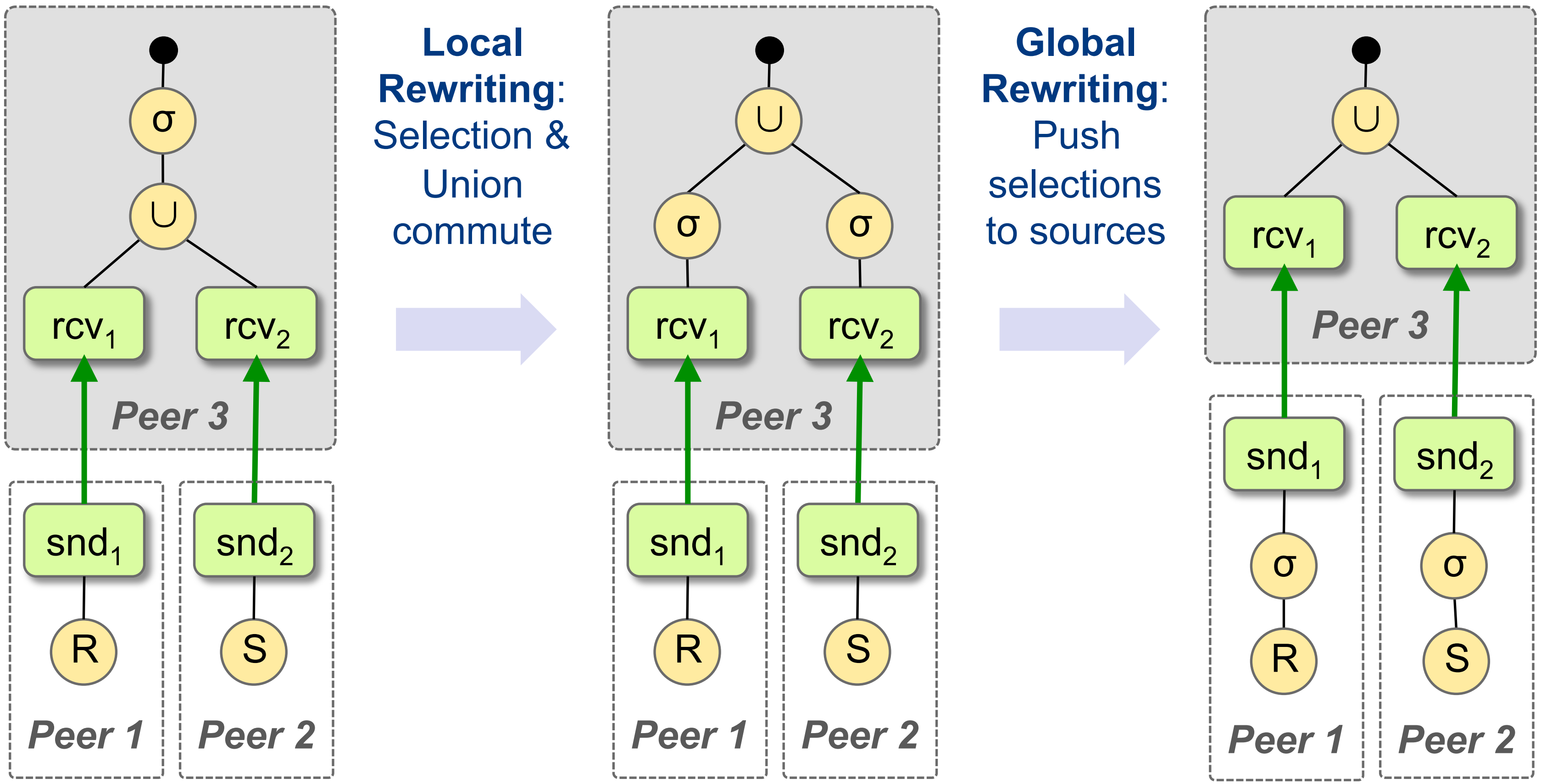
Equivalence of Distributed Systems with Queries and Communication

Serge Abiteboul INRIA Saclay & ENS Cachan
Balder ten Cate UC Santa Cruz
Yannis Katsis INRIA Saclay & ENS Cachan

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

Motivation: Distributed query optimization

e.g. p3 asks for σ (R@p1 ∪ S@p2)



Problem: Equivalence of distributed systems

When do two systems yield the same result?

Formalization of the problem

Modeling a distributed system: Active XML

An AXML System is a set of

- **finite**, **unranked**, **labeled** trees [XML docs]
  that are **unordered**
- that include **monotone queries** [TPQs with joins]
- and **send** and **receive** services for modeling communication

What kind of trees?

What kind of trees?

Passive nodes root a b

Annotated with labels

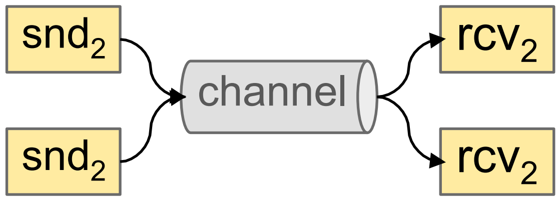
Query nodes q

Annotated with queries

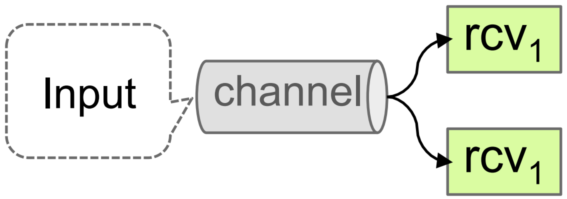
Send/Receive nodes snd_2 rcv_2 rcv_1

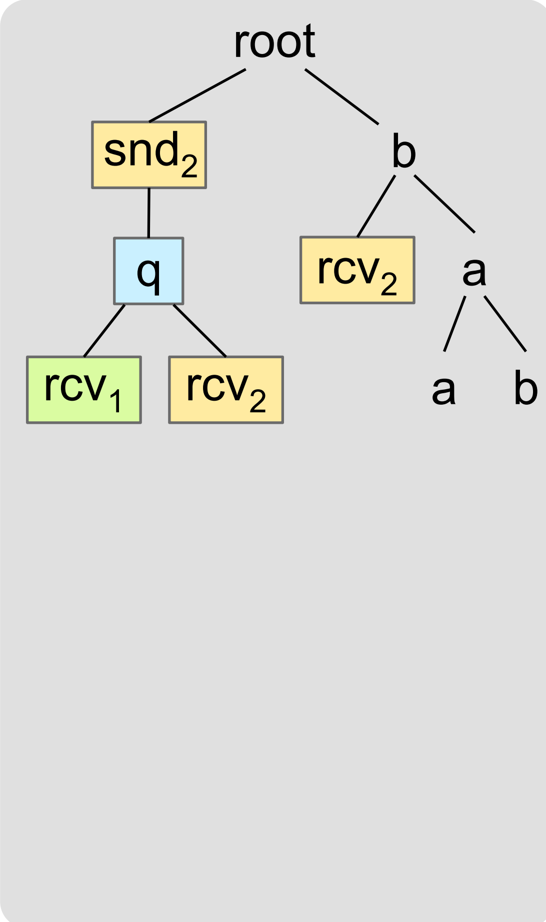
Annotated with channel ids

Internal channel



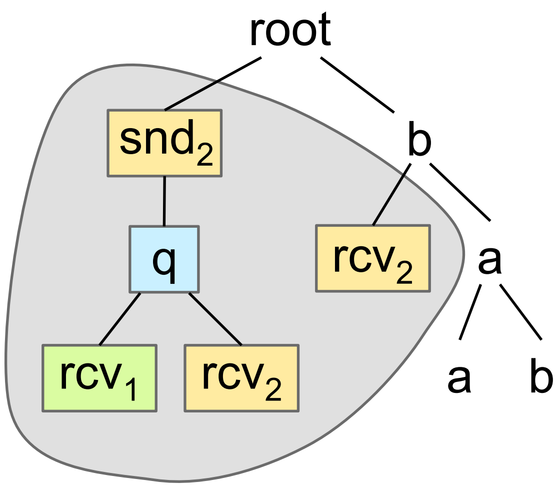
Input channel (no snd)





Parenthesis: Snapshot of a system

Contains only the passive data



- Queries are evaluated on snapshots
- Only passive data are sent

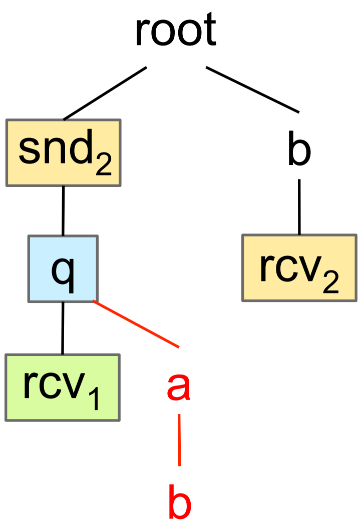
Evolution of a system

A system can evolve by activating:

- a **query** node
- a **send/receive** node on an **internal channel**
- a **receive** node on an **input channel**

Evolution Step: Receive on Input Channel

Model external inputs (seen as black boxes)

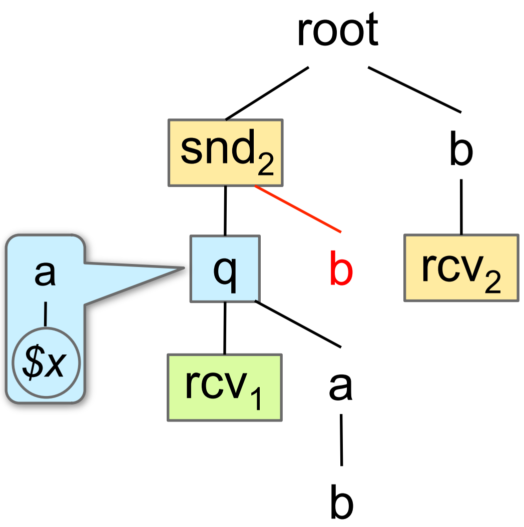


- Receive a forest from the input
- Place it as sibling of the rcv node

Non-deterministic

Evolution Step: Evaluate Query

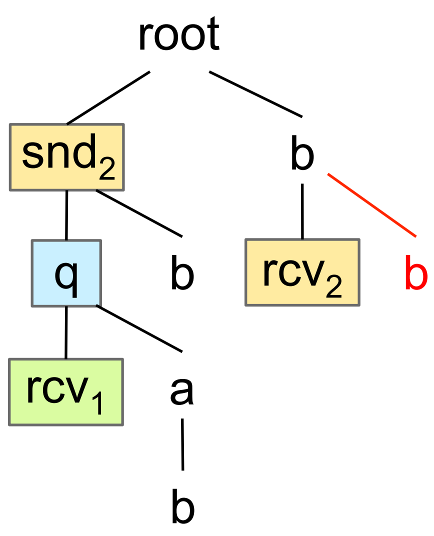
Model query evaluation



- Evaluate *q* on the snapshot of the descendants
- Place result as siblings of the query node

Evolution Step: Send on Internal Channel

Model communication between peers



- Take a snapshot of the descendants of the snd node
- Copy it as sibling of *all* rcv nodes of the same channel

Global effect of the example:

A query is applied to the external input and the result placed under r/b

Run of a system

A sequence of evolution steps $\underbrace{I = I_1 \rightarrow I_2 \ldots I_{n-1} \rightarrow I_n = I'}_{I \rightarrow^* I'}$

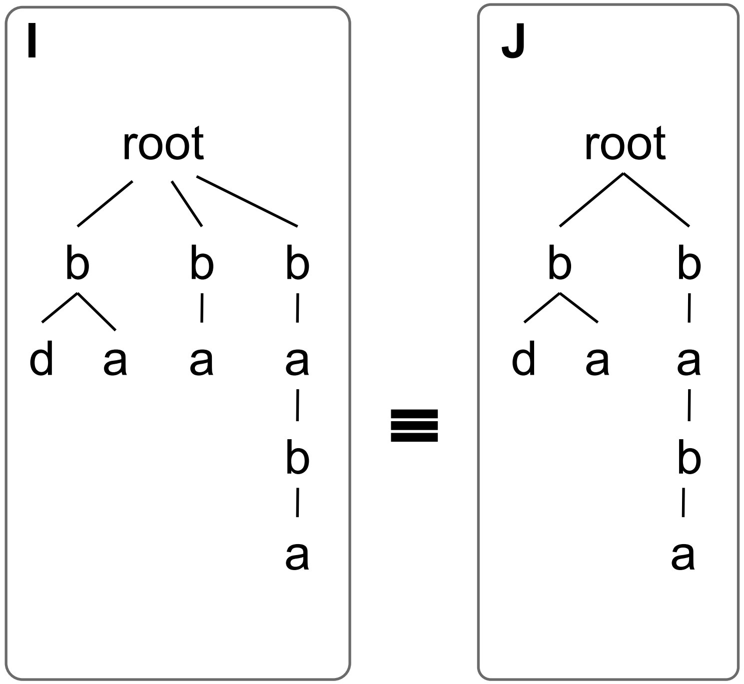
Parenthesis: Homomorphism

Homomorphism from I to J: I < J

- J has more information than I

Homomorphic equivalence: I ≡ J

- if I < J and J < I



Parenthesis: Homomorphism

Homomorphism from I to J: I < J

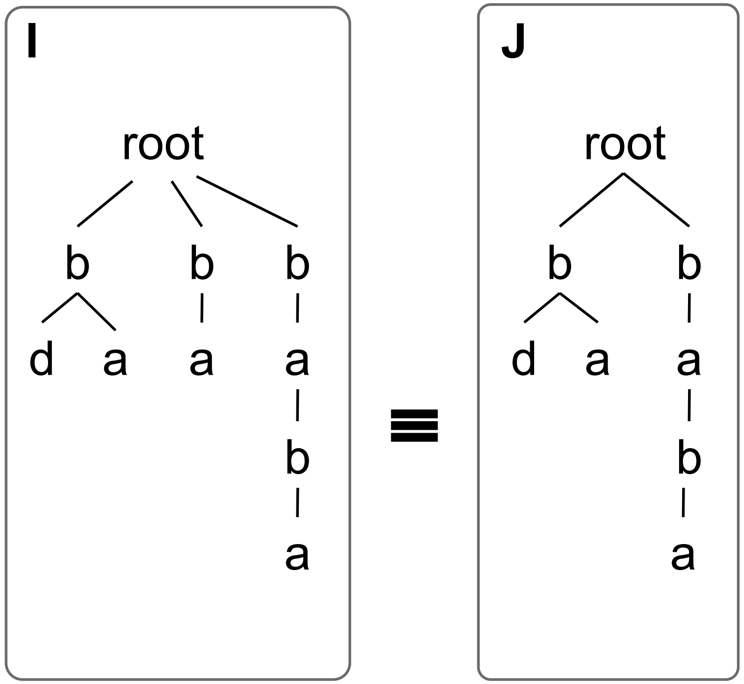
- J has more information than I

Homomorphic equivalence: I ≡ J

- if I < J and J < I

Reduced tree: A tree s.t. there does not exist a strict subtree J with the same root such that I ≡ J

- Undistinguishable for our query languages
- We consider only reduced trees



Semantics of Equivalence

Two systems I, J are *equivalent*

if for each run I $\rightarrow^*$ I’, there exists a run J $\rightarrow^*$ J’ with snapshot(I’) < snapshot(J’)

and vice versa

Main contribution:
Equivalence problem for AXML systems

	No query	TPQ	TPQ with XPath joins	TPQ with joins	TPQ with constructor
No input	PTIME	PTIME	PTIME	Hard	Undecidable
Input	PTIME	Hard	Hard	?	Undecidable

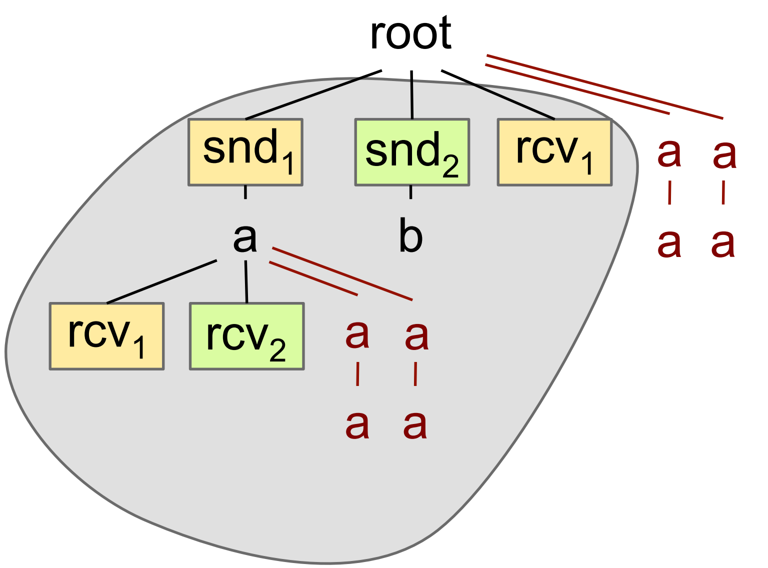
Complexity increases with:

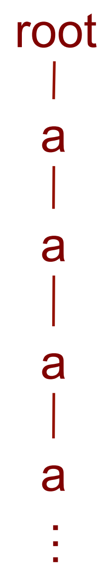
- ➔ richer query language
- ↓ input

Query-free & input-free systems

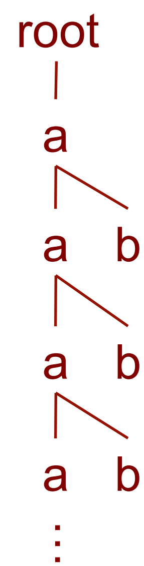
So many runs: Which one to look at?

A system has many different runs





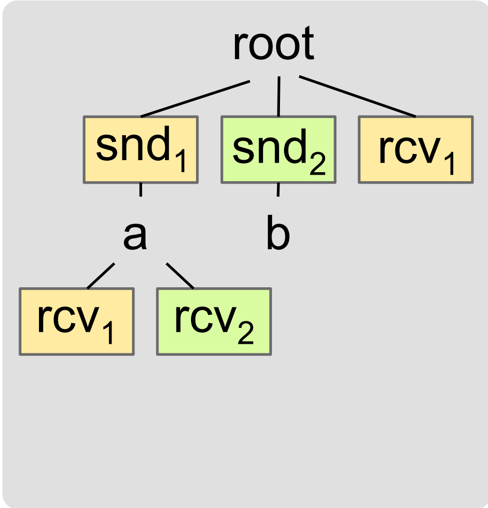
snapshot of **unfair** run

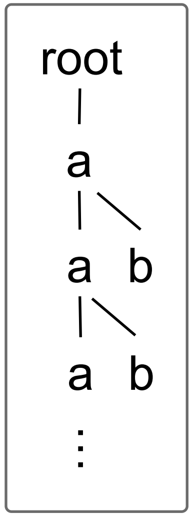


snapshot of **fair** run

Look at the Limit

It captures the result of an infinite fair run





I* is a *limit* of I if

- If I $\rightarrow^*$ I' then snapshot(I') < I*
- For each finite prefix J* of I*, there is I', I $\rightarrow^*$ I' and J* < snapshot(I')

Thm: Two systems I and J with finitely branching limits I*, J* are equivalent iff I* ≡ J*

A finite representation of a limit

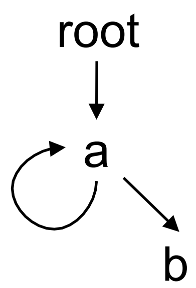
A finite graph whose unraveling is a finitely branching limit of I

Constructing the representation of a limit

❶ For each rcv_i: Add an edge from the parent of rcv_i to all children of all snd_i

❷ Remove all snd/rcv nodes and the nodes that are unreachable from the root

e.g.



Results for query-free & input-free systems

Decision procedure for equivalence

- Construct graph(I) & graph(J)
- Check whether they yield the same unravelings by checking simulation between the two graphs

Can be done in PTIME

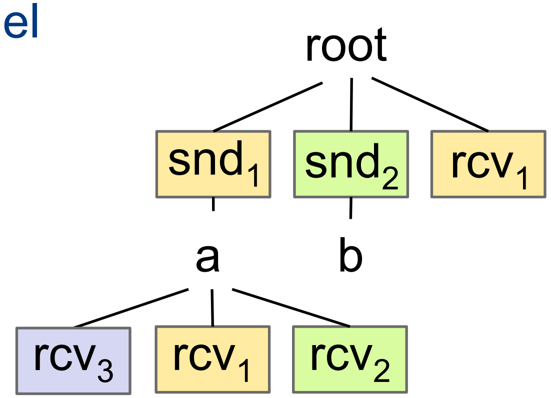
Results for query-free systems with inputs

Same complexity

- Replace each receive from an input channel by a fresh passive node
- Reuse previous procedure

Why does this work?

- Without queries “one cannot look inside the input”



Input-free systems with queries

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

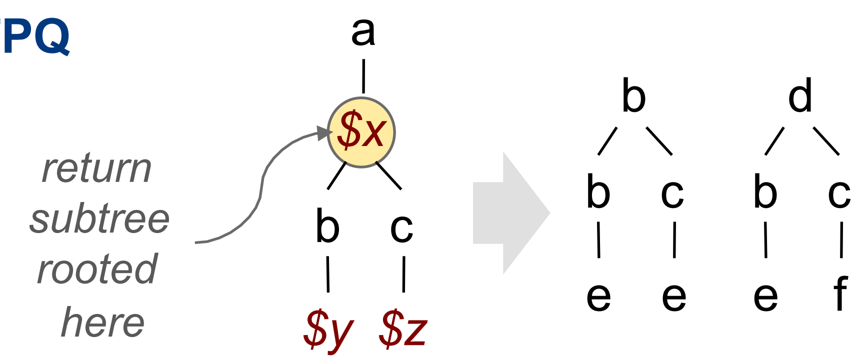
INRIA

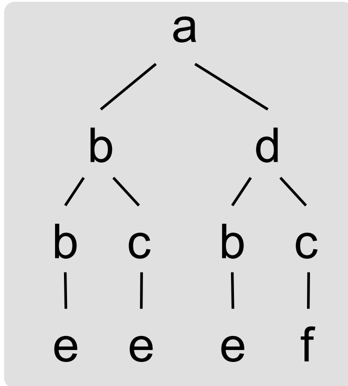
centre de recherche
SACLAY – ÎLE-DE-FRANCE

Query Languages

Classes of tree pattern queries

TPQ





TPQ with XPath-joins
(aka downward navigational XPath with path equality)

TPQ with arbitrary joins

TPQ with node constructors

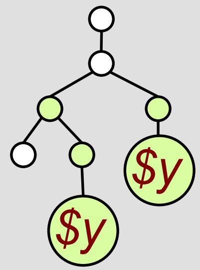
Undecidable

TPQ with XPath-joins

Downward navigational XPath with path equality

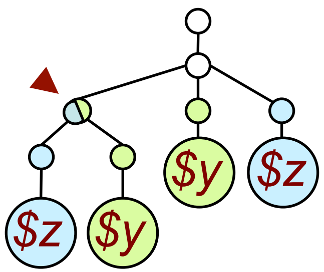
Intermediary of a pair of joining nodes

Any node in the shortest path between the nodes apart from their least common ancestor

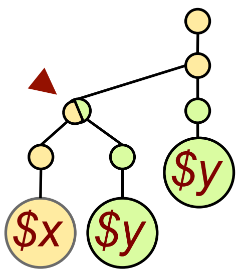


In a TPQ with XPath-joins:

- ✗ No node is an intermediary of 2 pairs



- ✗ No node in the path from the root to the result node is an intermediary



Results for input-free systems

Main idea

- Construct graph(I) by evaluating a datalog program with relations *child(m, n)* and *label(n, a)*
- Compare graphs through simulation

Complexity

- $P_{||}^{NP}$: Deterministic PTIME with parallel access to an NP oracle

Restricted to XPath-joins

- PTIME (due to bounded tree-width)

Systems with queries and inputs

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
SACLAY – ÎLE-DE-FRANCE

Equivalence of systems with queries & inputs

The problem is still open

Special cases:

Input is over a finite alphabet: Decidable

- Model limit as a monadic datalog program & check equivalence of two such programs [GottlobKoch04]

TPQs with XPath-joins: 3EXPTIME

- Simplify system by pushing queries directly over input channels
- Simplification requires more expressive query language: Regular TPQs with XPath-joins
- Use [Figueira09] to check equivalence of such queries

Axiomatization for query-free systems with inputs

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
SACLAY – ÎLE-DE-FRANCE

Axiomatization

Axiom scheme consisting of 8 axioms that

- **normalize** the system (moving send nodes directly below the root)
- **minimize** the system (removing inaccessible channels or channels that simulate each other)

Thm: Query-free systems I and J are equivalent iff one can rewrite I to J using the axioms

Conclusion

Foundations of distributed query processing

Starting point: AXML algebra

Here: Basis of a theory

Understand the impact of input, query language & other features such as constructors on equivalence

Open questions

- Decidability of equivalence for systems with inputs and queries
- Axiomatization of system with queries

Study the non-monotone case

- Synchronization issues

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
SACLAY – ÎLE-DE-FRANCE