Peer-to-Peer Schema Mediation

Bernstein, et al. “Data Management for Peer-to-Peer Computing: A Vision”, WebDB 2002

Halevy, et al. “Schema Mediation for Peer Data Management Systems”, ICDE 2003

Zachary G. Ives
University of Pennsylvania

CIS 650, Spring 2003

April 16, 2003

Administrivia

- Project status reports due today
  - Should have both a timeline and a plan for validating that your system does something useful
  - (If not, you need to work on that ASAP!)
- On 4/16 you need to have enough of the project working that:
  - You can talk about it for 5 minutes
  - You can answer my detailed questions about how/how well it works (i.e., it’s not vaporware!)

The Need for Schema Mediation

- In any company, collaboration, university, etc.:

  Different organizational units each have their own DBMS, schema, (partly overlapping) data, servers

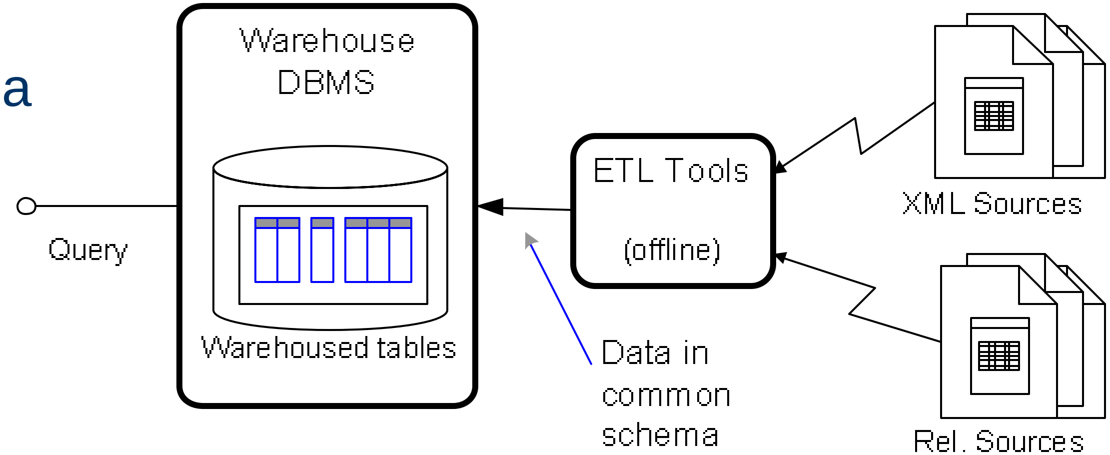
- Often important to get global view of the data *across* an organization
- May want to share data with *other* organizations, business partners, collaborators, customers, etc.

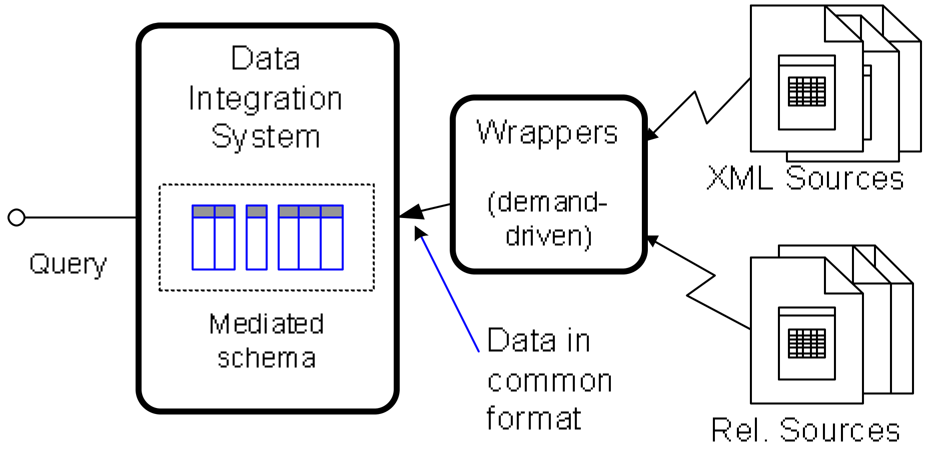
Problem: mediating (translating) between schemas

Approaches We've Seen to Schema Mediation

Data warehouse

- Design a single schema
  - Do physical DB design
- Map data into warehouse schema
- Periodically update warehouse





Virtual data integration (EII)

- Design mediated schema
- Map sources to mediated schema
- Queries are rewritten and answered on demand from sources

A Single Centralized Schema is a Bottleneck!

Challenging to form a single schema for all domain data

- People don’t agree on how concepts should be represented
- Data warehouse: physical design is a strong consideration
- Mediated schema very different from original users’ schemas

Mappings may be challenging to create, and do not leverage work of previous source mappings

Each source gets mapped to mediated schema separately

Difficult to evolve this single schema as needs change

- May “break” existing queries
- Must build consensus for any schema changes

What People Often Do…

Create ad hoc custom mappings between source pairs

- Define some intermediary schema
- Use custom code to export one source's data
- Import that into the opposite source

Easily extensible – no need to agree on single schema!

Disadvantages:

- Point-to-point: $O(n^2)$ translators may be necessary
- Often requires custom code, batch updates
- Need to be careful to distinguish between local extensional data and global domain data (what does a table represent?)
  - Separate between books at amazon.com and "books in general"

One Solution – The Local Relational Model: Bernstein et al.

- A "vision paper" (not yet an implementation) from U. Trento, U. Toronto
- "Coordination formulas" between different peers' relations:

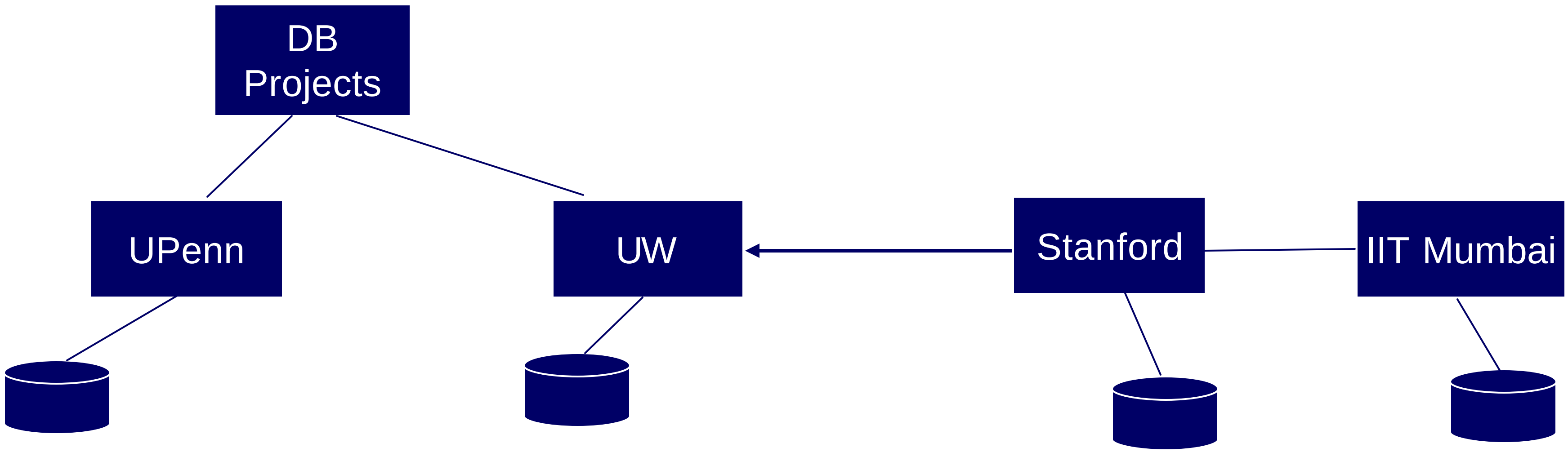
$$\forall fn \forall \ln \forall pn \forall sex \forall pr.(DavisDB:Patient(1234, fn, \ln, pn, sex, pr) \rightarrow$$

$$TGHDB:\exists tghid \exists n \exists a.(Patient(tghid, 1234, n, sex, a, Davis, pr) \wedge n = concat(fn, \ln)))$$

These define how to import data from one source into another

- Every time a data source is updated, its effects get propagated
- No distinction between global and local concepts – all data is, by default, imported into the same tables
- Contrast with the main paper for today…

Peer Data Management: Decentralized Mediation for Ad Hoc Extensibility



Data integration: 1 mediated schema, *m* mappings to sources

Peer data management system (PDMS):

- *n* mediated "peer schemas," as few as (*n* - 1) mappings between them – evaluated transitively
- *m* mappings to sources

Peer-to-Peer at both Logical and Architectural Levels

A "logical" peer-to-peer model:

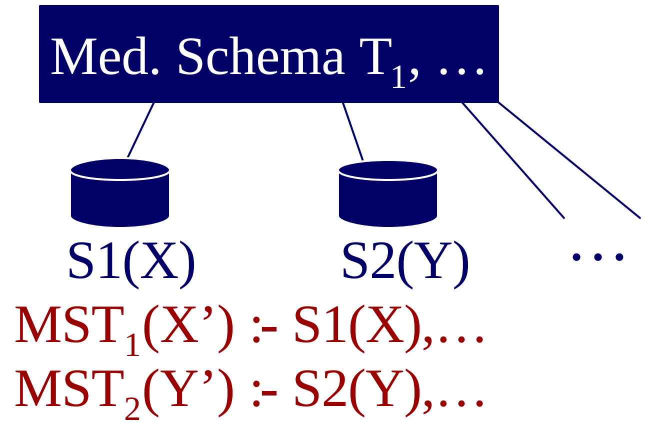
Every participant can contribute:

- Extensional data
- Mappings between schemas
- Computation (query answering) and caching

Mapping Formalisms from Data Integration

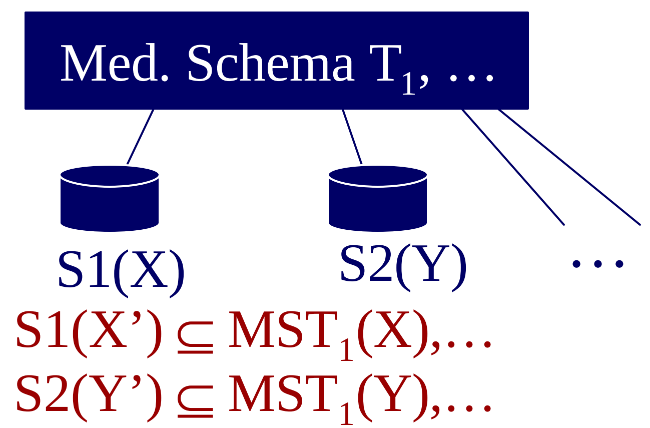
GAV: mediated relations as views over sources

- Easy to rewrite queries: unfold them using view definitions



LAV: sources as views over mediated relations

- More challenging to rewrite queries: answering queries using views (e.g., MiniCon [Pottinger & Levy 00])
- More flexible in representing source properties



Answering Queries in a PDMS: Transitively Evaluating Mappings

Mappings in a PDMS are a generalization of LAV, GAV techniques (GLAV):

- *Query over schema 1 = Query over schema 2 (where possible)*
  But there are lots of limitations on when this is decidable!
- Requires unfolding: p(X) :- v1(X', Y), v2(Y, Z), …
- Requires AQUV: p(X, Y), p(Y, Z) :- v(X', Y')

Start with schema being queried

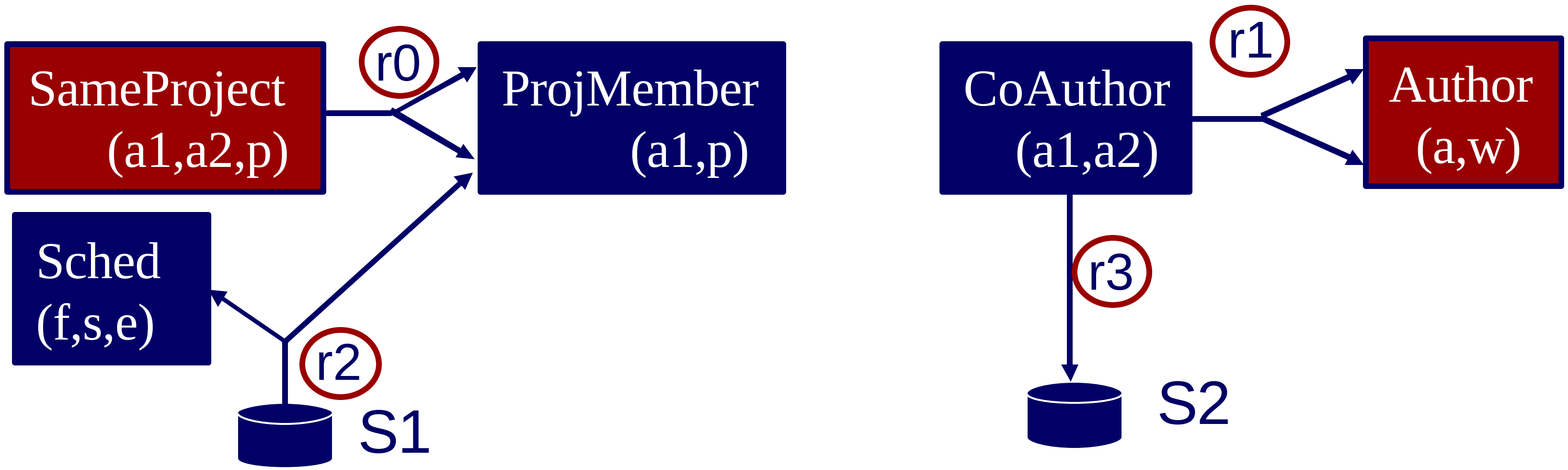
- Look up mappings to neighbors; expand
- Continue iteratively until queries only over sources

We use a rule-goal "tree" to expand the mappings

- Extend some of the ideas of MiniCon to avoid unnecessary expansions
- Challenges to avoid redundancy – see paper for optimizations

Example of Query Answering

Query: Q(a1, a2) :- SameProject(a1,a2,p), Author(a1,w), Author(a2,w)



Mappings between peers' schemas:

r0: SameProject(a1,a2,p) :- ProjMember(a1,p), ProjMember(a2,p)

r1: CoAuthor(a1,a2) $\subseteq$ Author(a1,w), Author(a2,w)

Mappings to data sources:

r2: S1(a,p,s) $\subseteq$ ProjMember(a,p), Sched(f,s,end)

r3: CoAuthor(f1,f2) :- S2(f1,f2)

Example Rule-Goal Tree Expansion

q: Q(a1, a2) :- SameProject(a1,a2,p), Author(a1,w), Author(a2,w)

Example Rule-Goal Tree Expansion

q: Q(a1, a2) :- SameProject(a1,a2,p), Author(a1,w), Author(a2,w)

Example Rule-Goal Tree Expansion

q: Q(a1, a2) :- SameProject(a1,a2,p), Author(a1,w), Author(a2,w)

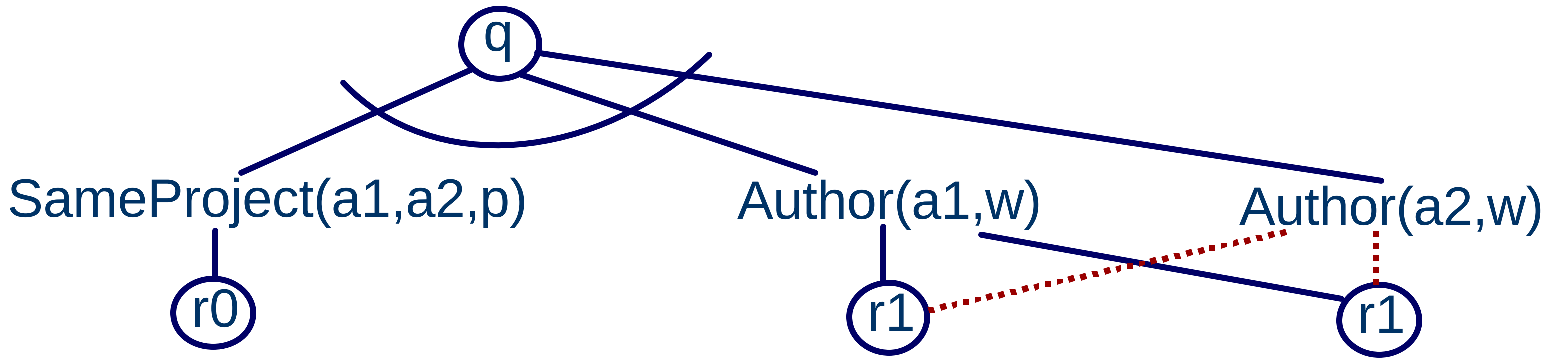
Mappings between peers' schemas:

r0: SameProject(a1,a2,p) :- ProjMember(a1,p), ProjMember(a2,p)

r1: CoAuthor(a1,a2) $\subseteq$ Author(a1,w), Author(a2,w)

Example Rule-Goal Tree Expansion

q: Q(a1, a2) :- SameProject(a1,a2,p), Author(a1,w), Author(a2,w)



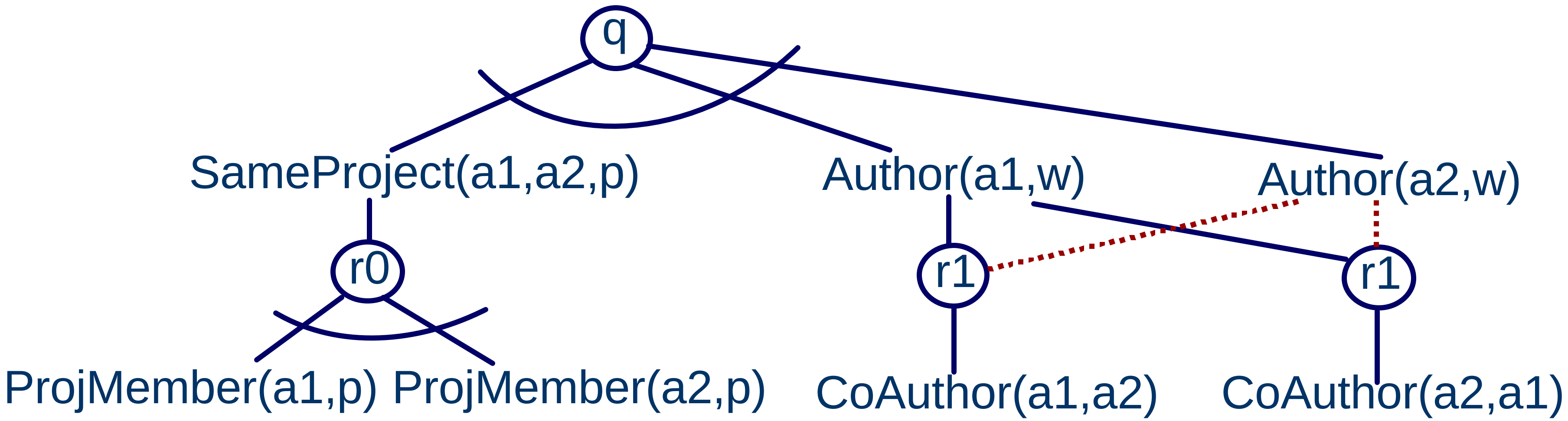
Mappings between peers' schemas:

r0: SameProject(a1,a2,p) :- ProjMember(a1,p), ProjMember(a2,p)

r1: CoAuthor(a1,a2) $\subseteq$ Author(a1,w), Author(a2,w)

Example Rule-Goal Tree Expansion

q: Q(a1, a2) :- SameProject(a1,a2,p), Author(a1,w), Author(a2,w)



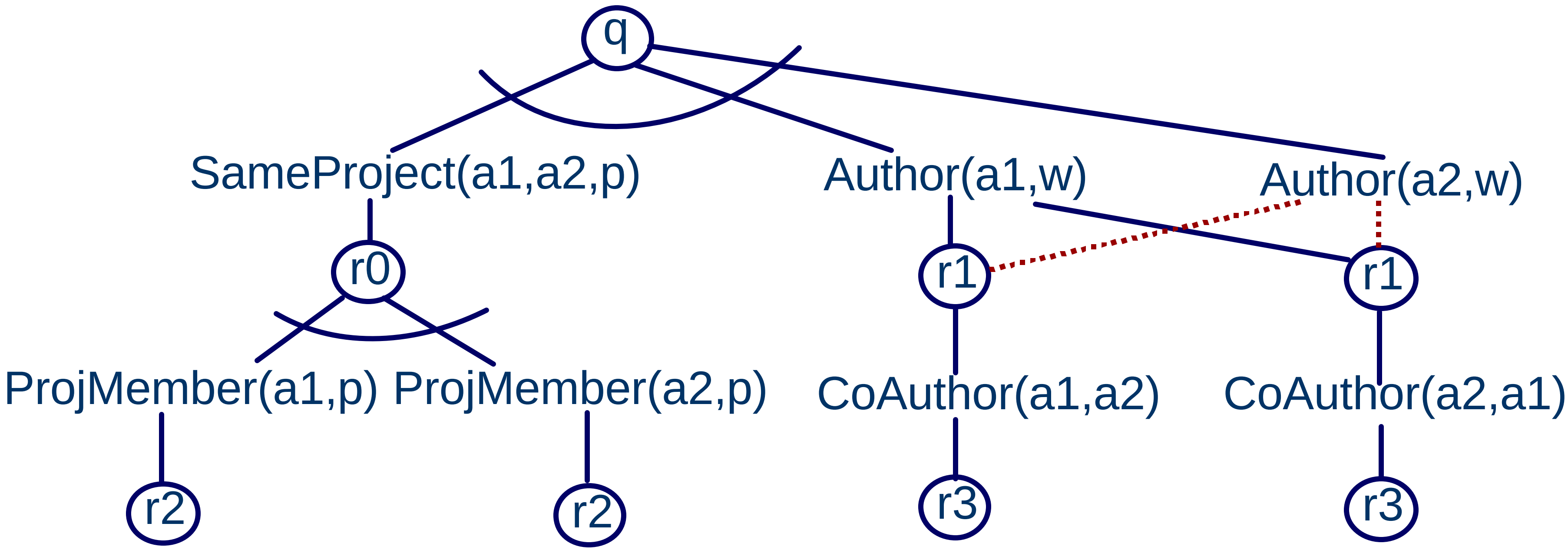
Mappings to data sources:

r2: S1(a,p,s) $\subseteq$ ProjMember(a,p), Sched(a,s,end)

r3: CoAuthor(f1,f2) = S2(f1,f2)

Example Rule-Goal Tree Expansion

q: Q(a1, a2) :- SameProject(a1,a2,p), Author(a1,w), Author(a2,w)



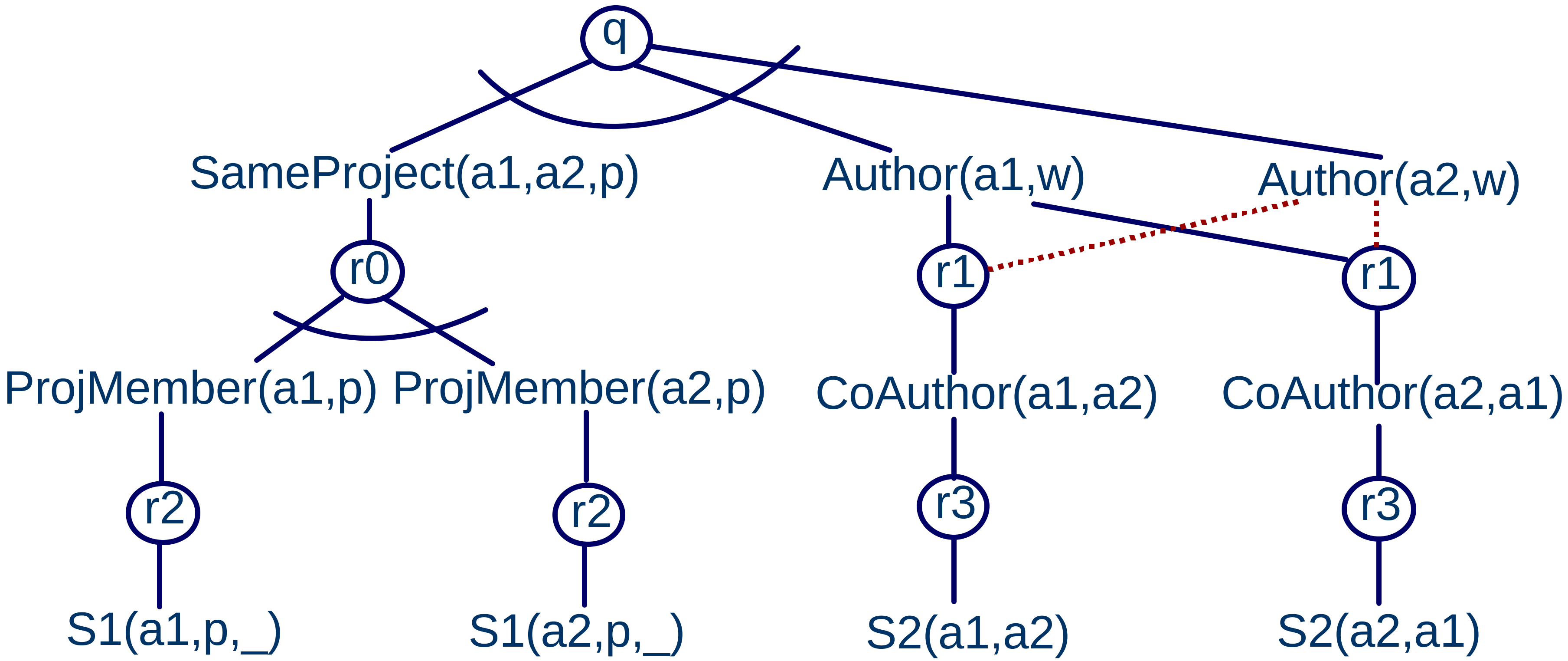
Mappings to data sources:

r2: S1(a,p,s) $\subseteq$ ProjMember(a,p), Sched(a,s,end)

r3: CoAuthor(f1,f2) = S2(f1,f2)

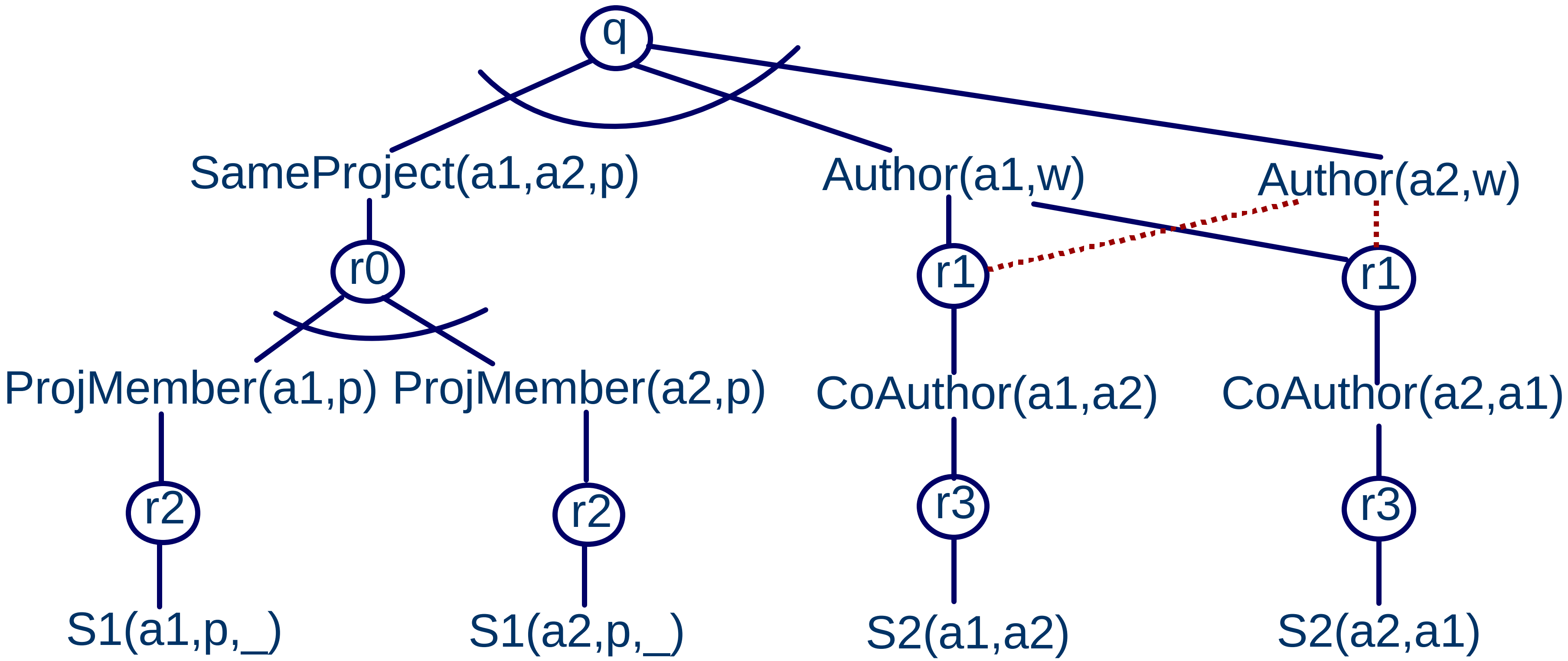
Example Rule-Goal Tree Expansion

q: Q(a1, a2) :- SameProject(a1,a2,p), Author(a1,w), Author(a2,w)



Example Rule-Goal Tree Expansion

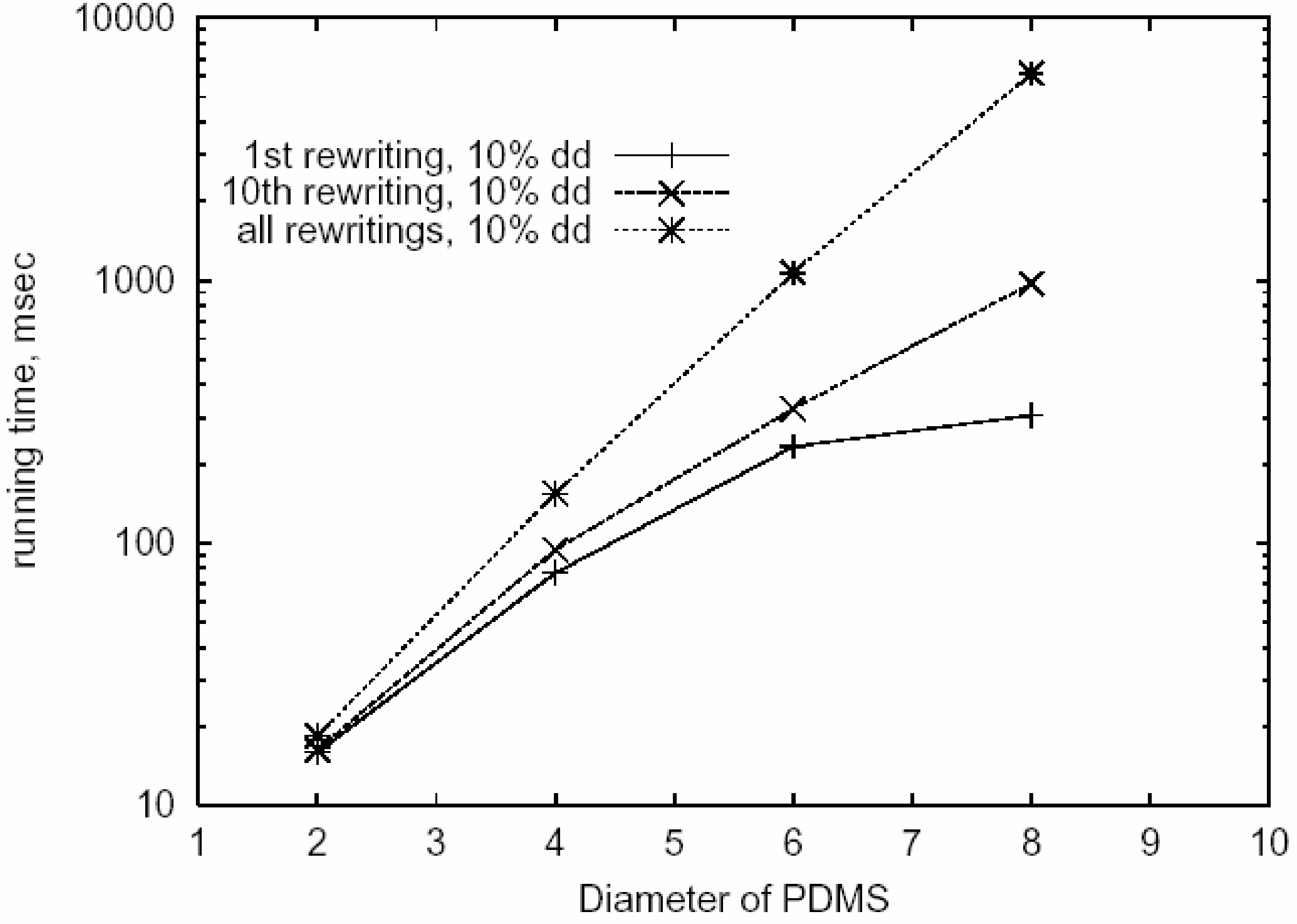
q: Q(a1, a2) :- SameProject(a1,a2,p), Author(a1,w), Author(a2,w)



Q'(a1,a2) :- S1(a1,p,_), S1(a2,p,_), S2(a1,a2)
∪ S1(a1,p,_), S1(a2,p,_), S2(a2,a1)

Algorithm Scales Well to Large-Diameter PDMSs

- Randomly generated peers, definitions (simulated infrastructure)
- Relatively unoptimized Java implementation



Schema Mediation: The Core of Peer Data Management

Sharing data across schemas is a key problem today

- PDMS approach is much more flexible and extensible
- Composition of mappings leverages others' work

One step towards a larger vision:

- Much of the power of the "semantic web" but scalable
  - We'll talk about the semantic web in a few weeks
- Scalable, extensible P2P architecture for data sharing

Further Ongoing Work

Applying to real bioinformatics applications!

Caching and replication

- Intelligent placement of data
- Updating caches [Mork et al]

Studying mappings:

- Information loss and approximate mappings
- Composition [Madhavan & Halevy]
- Automatically learning mappings [Doan et al]

Reconciling updates across mappings