Complete Answer Aggregates for Answer Mappings to Sequence, Tree and Graph Databases

Holger Meuss

Center for Information and Language Processing

University of Munich

December 2, 1999

Joint work with Klaus Schulz

My Supporters

- Interdisciplinary fellowship SIL (Speech, Information, Logic) at University of Munich
- Center for Information and Language Processing (Franz Guenthner, Klaus Schulz):
  - CISLEX
  - text / web indices (German webpages for altavista.de)
  - structured document retrieval
- Chair for Programming and Modeling Languages (François Bry):
  - logic programming
  - constraint programming
  - XML and semistructured data

Where do I come from?

Structured Documents: Documents with logical structure, e.g. SGML or XML.

Structured document retrieval in the overlap of Information Retrieval and Database Systems.

Tree Matching formalizes queries and databases as trees, answers as mappings (Kilpeläinen 93).

Problems with combinatorial explosion.

Solution for Tree Matching: Complete Answer Aggregates.

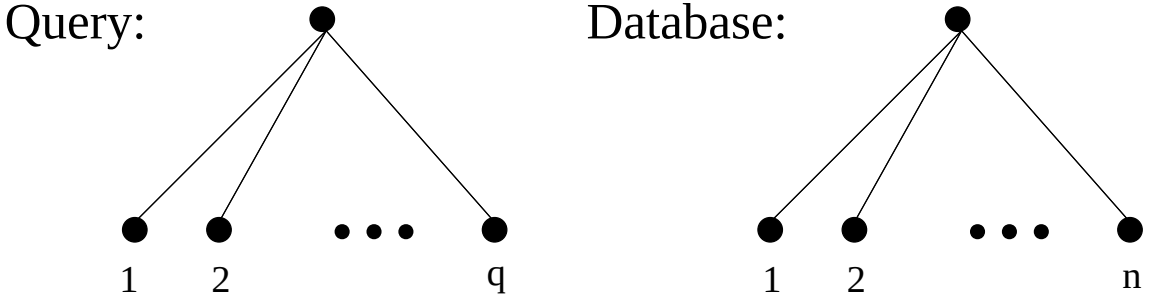
Useful for general graph databases?

Graph Databases

Application domains: medicine, biology, multimedia data, semistructured data, etc.

Query and database have graph structure, answers are mappings.

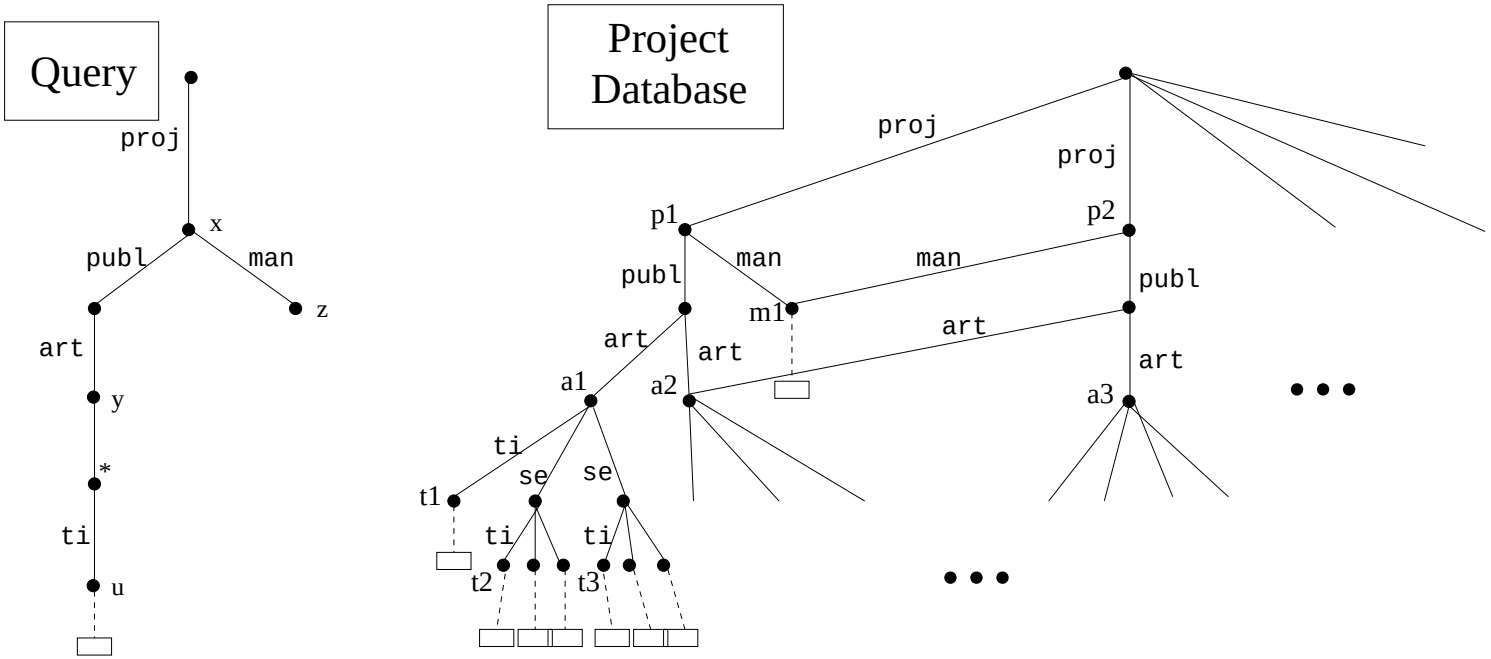
Combinatorial explosion in the number of answers:



has n^q answers, where distinct answers share common nodes.

- high theoretical complexity in computation of answers
- unsatisfactory answers for the user

Example: Project Database



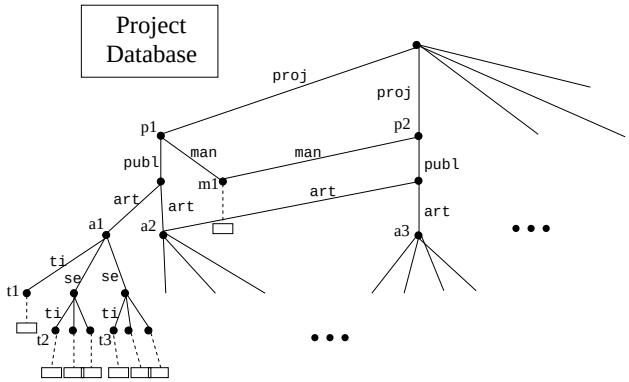

```
x -> p1, y -> a1, z -> m1, u -> t1
x -> p1, y -> a1, z -> m1, u -> t2
x -> p1, y -> a1, z -> m1, u -> t3
x -> p1, y -> a2, z -> m1, u -> t4
x -> p2, y -> a2, z -> m1, u -> t1
...
```

Goals of the Talk

- Point out problem of answer presentation in graph databases
- (Define a simplistic graph database and query model)
- Present a new, intensional answer concept: "Complete Answer Aggregate" (CAA)
- Discuss computation of CAAs
- Discuss relation between CAAs and arc-consistency algorithms in constraint networks
- Show use of CAAs for interactive analysis and modification

Database Model

Database: $\mathcal{D} = (N, E, L_N, L_E, \Lambda_N, \Lambda_E)$:



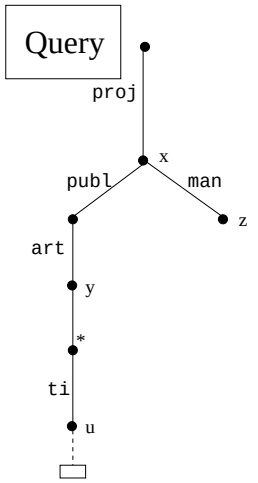
- Nodes N with edges $E \subseteq N \times N$
- Sets of labels: L_N for nodes and L_E for edges
- $\Lambda_N : N \to 2^{L_N}$ assigns set of labels to each node
- $\Lambda_E : E \to L_E$ partial function assigning labels to edges

$\mathcal{D}$ can be a *sequence*, *tree*, *DAG*, or *graph database* depending on the structure E imposes on N.

Query Model

Four classes of *atomic formulae*:

- $A(x)$ (labeling constraints),
- $x \to y$ (child constraints),
- $x \to_f y$ (f-child constraints),
- $x \to_+ y$ (descendant constraints)



for variables $x, y \in X$, $A \in L_N$, and $f \in L_E$.

Query Q: finite conjunction of atomic formulae.

Edge constraints $x \to_f y, x \to y, x \to_+ y$ denoted with $x \to_\_ y$.

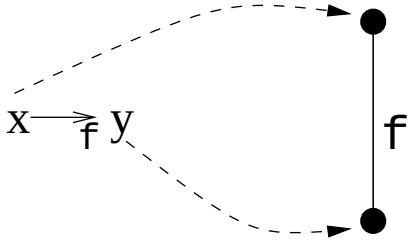
Queries as graphs: Variables correspond to nodes. Edge constraints correspond to edges.

Query Q is a *sequence*, *tree*, *DAG*, or *graph query* depending on the structure the edge constraints impose on variables X_Q in Q.

Answers

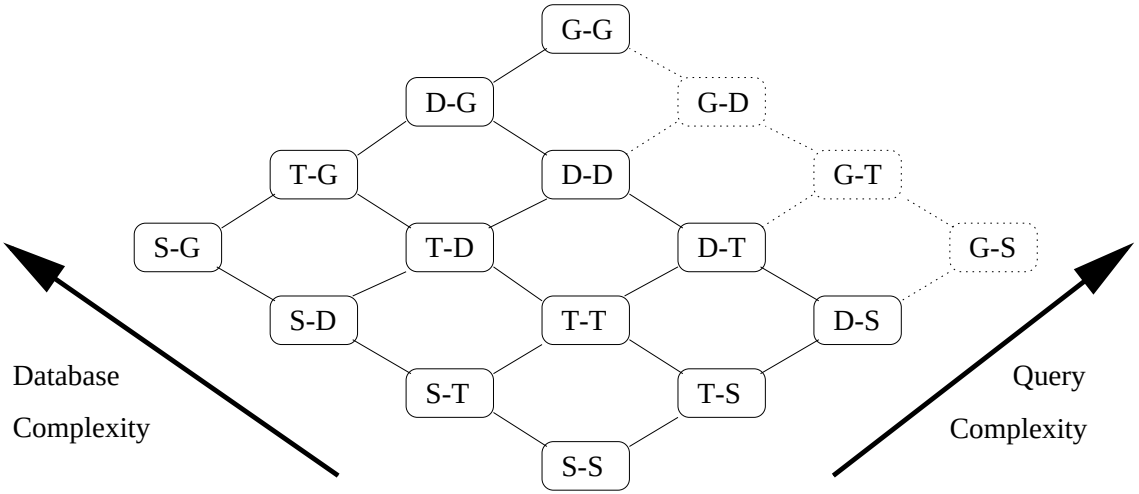
An *answer* to Q in $\mathcal{D}$ is a variable assignment $\nu : X_Q \to N$ so that:

- $A \in \Lambda_N(\nu(x))$ if Q contains labeling constraint $A(x)$
- $(\nu(x), \nu(y)) \in E$ if Q contains child constraint $x \to y$
- $(\nu(x), \nu(y)) \in E$ and $\Lambda_E(\nu(x), \nu(y)) = f$ if Q contains f-child constraint $x \to_f y$
- $(\nu(x), \nu(y)) \in E^+$ (trans. clos. of E) if Q contains descendant constraint $x \to_+ y$

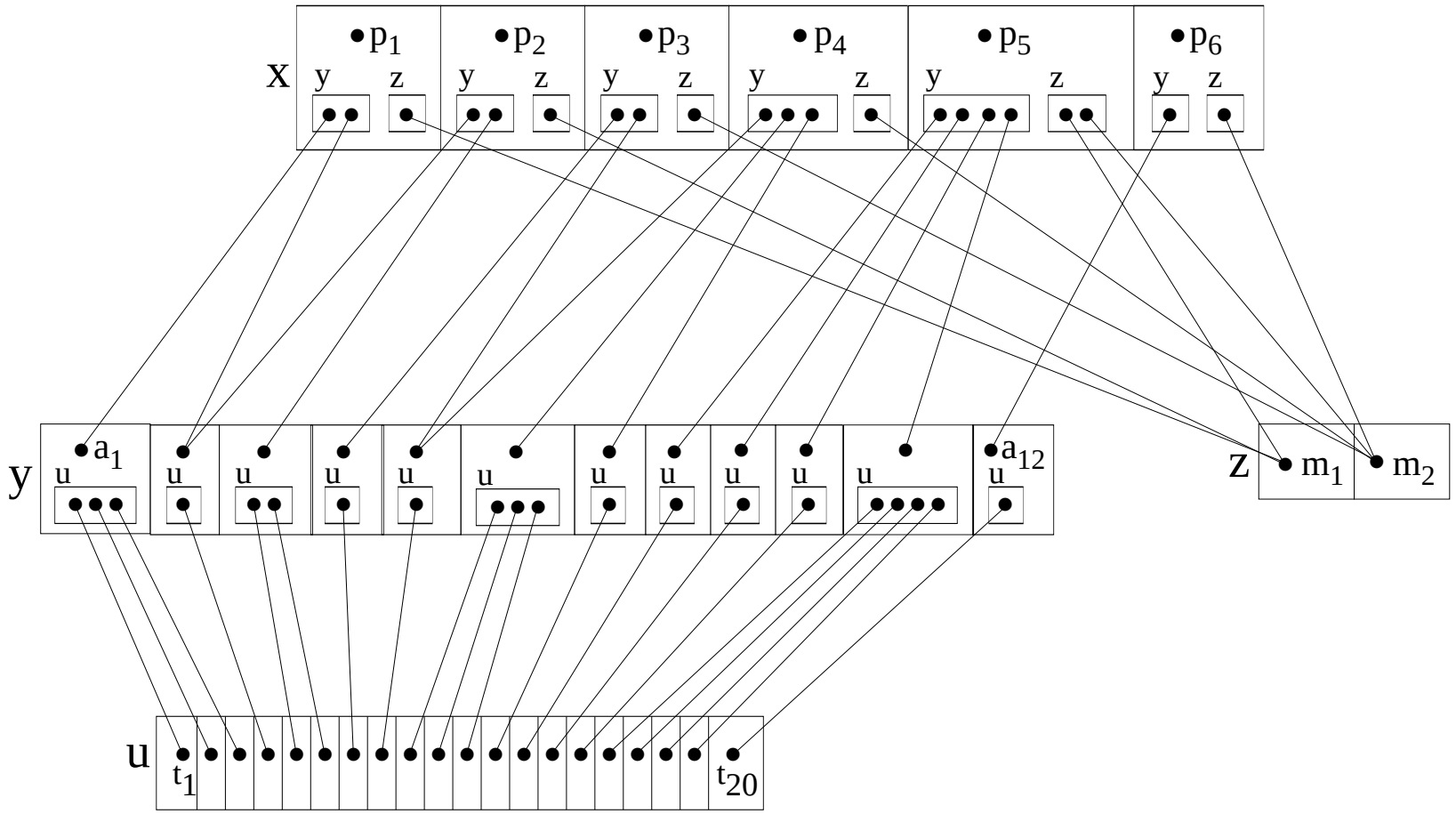


Hierarchy of Query Evaluation Problems

S-S evaluation problem $(Q, X_Q, \mathcal{D})$: Sequence query Q with variables X_Q and sequence database $\mathcal{D}$.
In the same way (S: sequence, T: tree, D: DAG, G: graph): S-T, S-D, S-G, T-S, T-T, T-D, T-G, D-S, D-T, D-D, D-G, G-G.



Complete Answer Aggregate (Example)

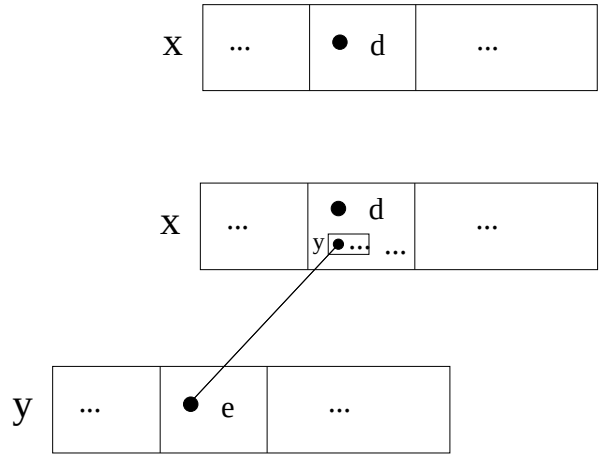


stores 29 answers.

Complete Answer Aggregates (CAAs)

Complete Answer Aggregate (CAA) (Dom, Π) for evaluation problem $(Q, X_Q, \mathcal{D})$ with $Dom : X_Q \to 2^N$ and $\Pi : \{(x, y) | Q \text{ contains } x \to\_ y\} \to 2^{N \times N}$ so that:

- $d \in Dom$ iff exists answer ν with $\nu(x) = d$
- $(d, e) \in \Pi(x, y)$ iff Q contains $x \to\_ y$ and exists answer ν with $\nu(x) = d$ and $\nu(y) = e$



$d \in Dom(x)$: *Target candidate* in *slot* x.

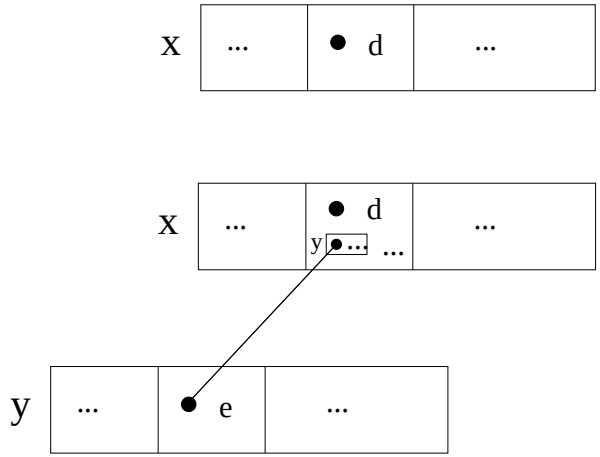
$(d, e) \in \Pi(x, y)$: *Link* from d to e.

For every evaluation problem there exists a unique CAA.

Instantiation of a CAA

An *instantiation* of a CAA (Dom, Π) is a mapping $\nu : X_Q \to N$ so that

- $\nu(x) \in Dom(x)$ and
- $(\nu(x), \nu(y)) \in \Pi(x, y)$ for every edge constraint $x \to\_ y$.



- Every instantiation of a CAA for $(Q, X_Q, \mathcal{D})$ is an answer to $(Q, X_Q, \mathcal{D})$ and vice versa.
- Every target candidate and every link contributes to an instantiation of a CAA.

Presenting CAAs

Present CAA to the user as a substitute for the set of all answers as a

- container for all answers in order to enumerate them,
- overview over the set of all answers with its structure and topology, making explicit dependencies,
- tool for interactive analysis, manipulation and query reformulation.

Size of CAAs

n: number of nodes in N, a maximal number of ancestors for a node $d \in N$, q number of variables in X_Q.

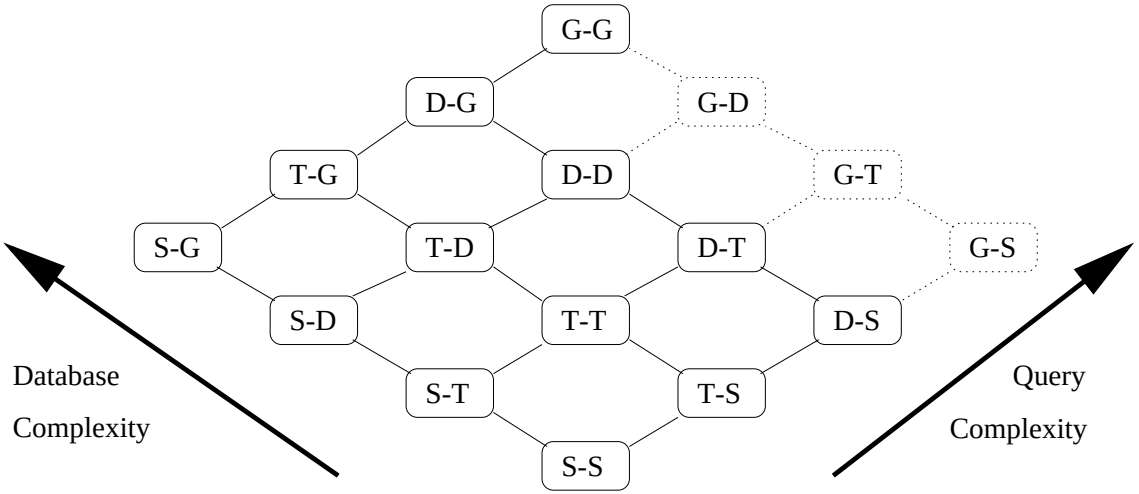
In general: n^q answers to query evaluation problem.

Size of an aggregate: #target candidates + #links

Theorem: The size of a CAA for an evaluation problem is of order $O(q \cdot n \cdot a)$.

Better bound for rigid queries, or non-recursive databases and completely labeled tree queries.

Computation of CAAs



- Sequence and tree queries (arbitrary database): recursive bottom-up computation in time and space $O(q \cdot n \cdot a)$.
- D-S evaluation problem: time $O(q \cdot e \cdot n^3 \cdot a)$ with help of arc-consistency techniques (e is the number of edges in Q).
- D-T, D-D, D-G, G-G: NP-complete. (Even decision problem whether there exists an answer is NP-complete.)

Tree Database Project

Implementation of ordered T-T evaluation problems for the application structured document retrieval:

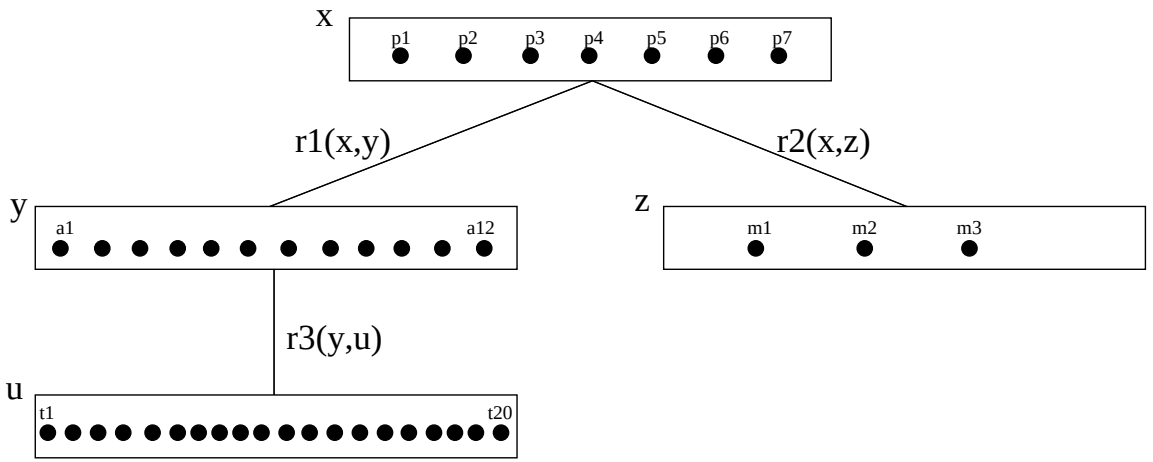
Algorithmic core for query evaluation finished.

Index structure, node database, graphical user interface: in progress.

Test databases:

- Collection of court decisions in SGML format
- Database of noun phrases

Constraint Networks (CNs)



Solution: consistent variable assignment

Arc-consistency: every node has a partner

Links implicit

Translation for tree queries:

Evaluation Problem $\Rightarrow$ CN $\rightsquigarrow$ arc-consistent CN $\Rightarrow$ CAA

For DAG and graph queries: arc-consistency is too weak!

CAA Analysis and Modification

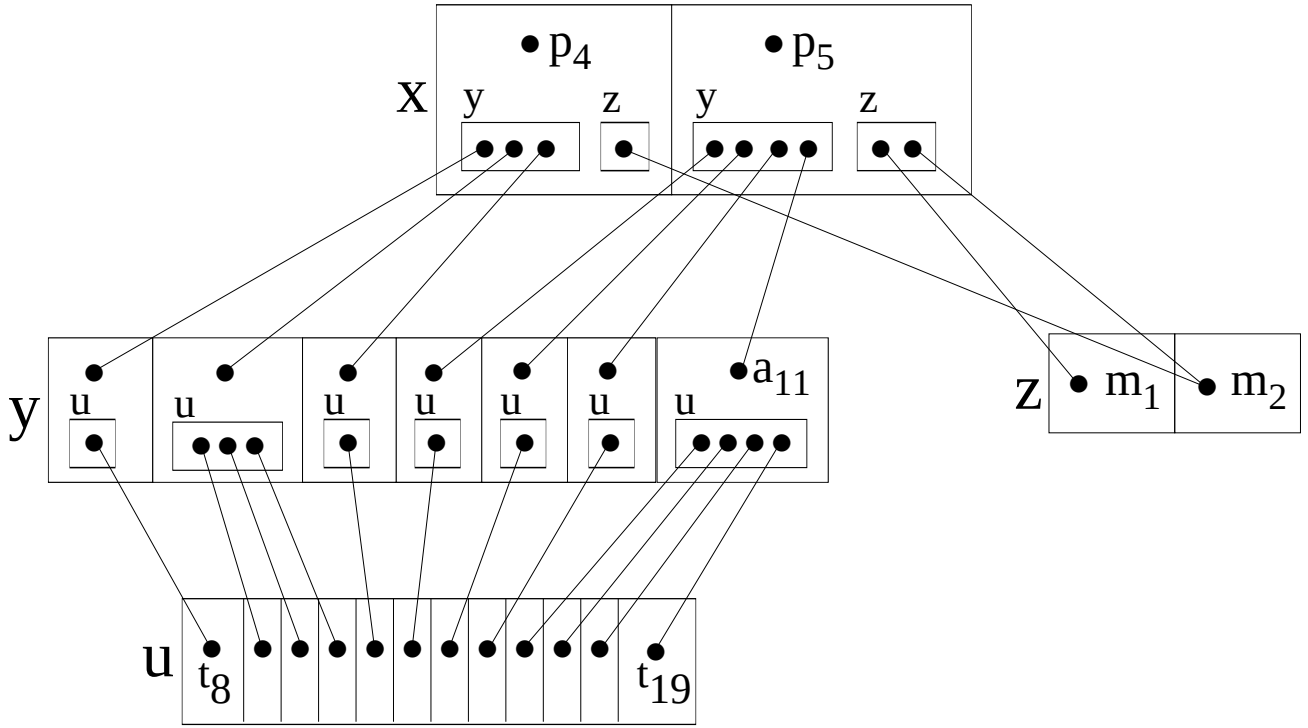
Use CAAs in a (possibly iterated) two-step retrieval process:

1. Define with a rich (many variables, few restrictions) query the "sphere of interest".
2. Analyse and modify in an interactive and graphical way the resulting answer aggregate in order to filter out interesting parts.

Direct use Output number of entities found for given variable. List entities. Display additional information like attributes or textual content of a chosen target candidate in another window.

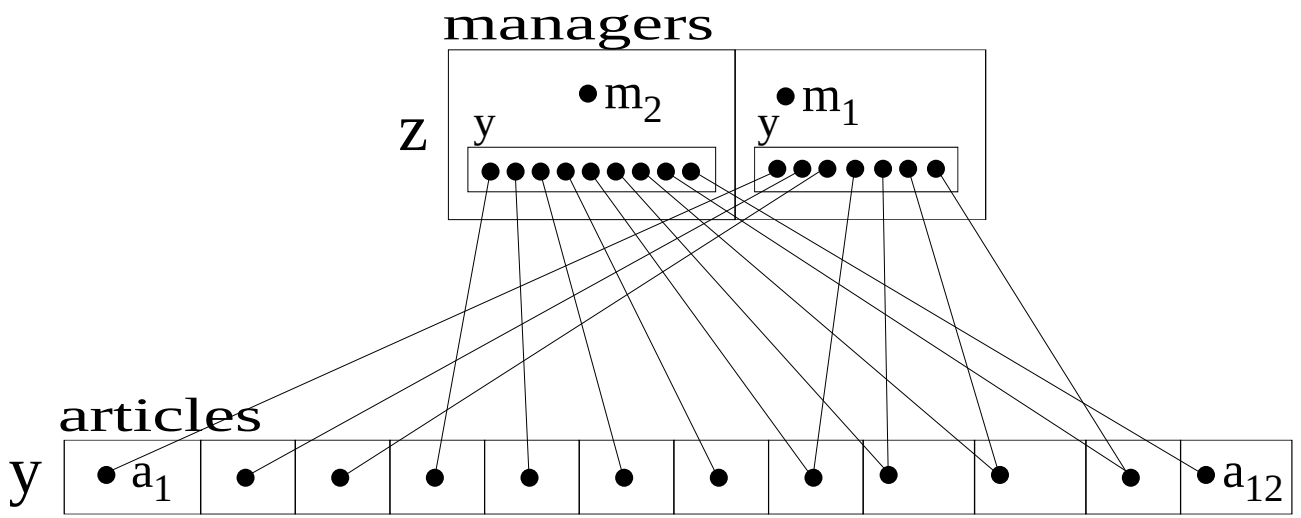
Count links *"Which manager has the largest number of projects?"* *"Which projects have published at least two articles?"*

Restricted view *"Display only projects with at least three relevant publications!"*



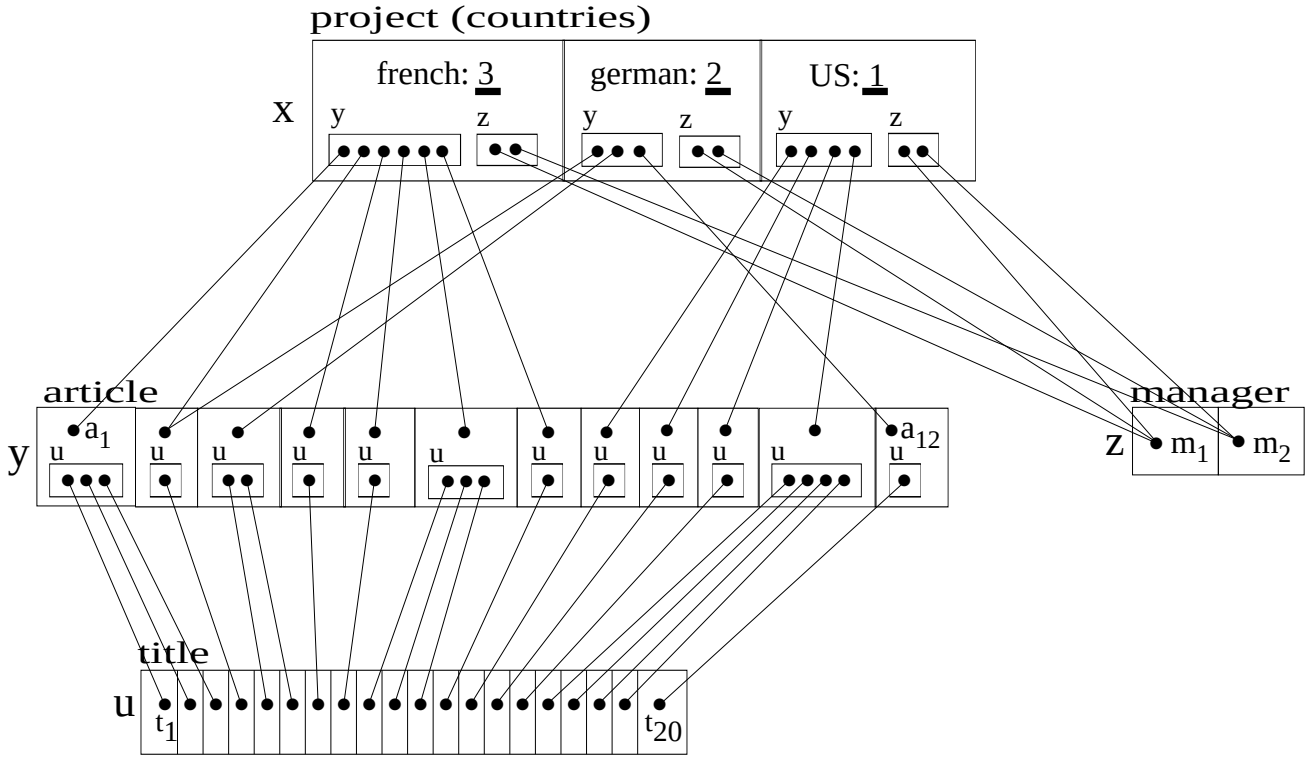
Projection *"Hide the project slot!"*

Ranking *"Sort managers w.r.t number of articles published in their projects!"*



"Sort articles w.r.t. date of publication!"

Equivalence Classes *"Merge target candidates with same attributes!"*



Controlled Enumeration *"Give me all answers with manager* m1*!"*

Problems and Open Questions

Problems with DAG and graph queries:

- NP-completeness for D-T, D-D, D-G, and G-G evaluation problems (even for decision problem).
- Enumeration of all answers in a CAA for DAG and graph queries requires backtracking (cf. Freuder 82).

(Do we need DAG and graph queries often?)

(Do we need the enumeration of CAAs?)

Are approximate answer aggregates (containing all answers and more) useful for DAG and graph queries? Computation?

Path constraints (e.g. Bertino and Kim 89, Abiteboul and Vianu 97) with regular expressions over edge labels fit into CAA framework. Computation?

Conclusion

CAAs as representation of the set of all answers

- are **general**, since they can be applied to formalisms based on answers as mappings
- provide an intensional **overview** over all answers
- are **intuitive** and easy to understand due to their graphic nature
- make **dependencies** between answers **explicit**
- **reduce size** of result presented to the user
- can be computed **efficiently** for sequence and tree queries
- provide rich field of **analysis** and **modification** techniques on a **visual** level