

# Scalable LAV Query Rewriting: Minicon, MCDSAT, GQR

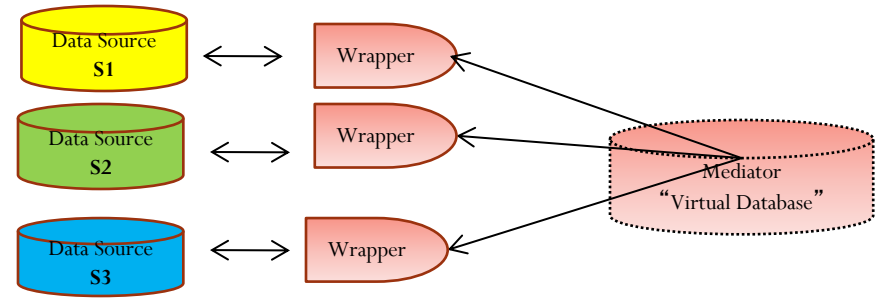
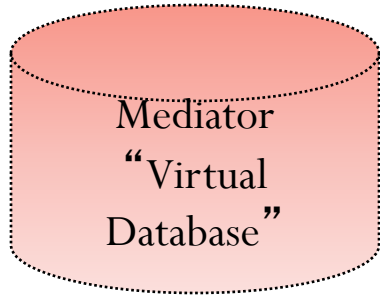
George Konstantinidis

USC Viterbi

School of Engineering  
*Information Sciences Institute*

# Data Integration

## Example: academic papers



**Schema:**

sameTopic		inSIGMOD	inVLDB
paper1	paper2	paper	paper

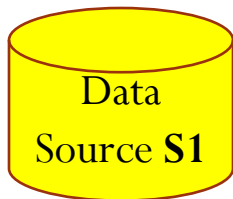
**Datalog:**

sameTopic(t1,t2)  
cites(p1,p2)  
inSIGMOD(p)  
inVLDB(p)

cites	
paper1	paper2

**Query:**

$Q(x,y) :- \text{sameTopic}(x,y), \text{cites}(x,y), \text{cites}(y,x)$

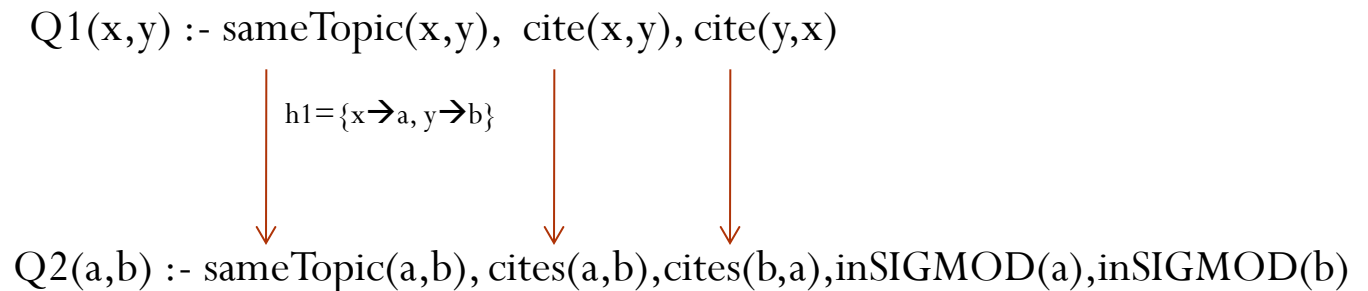


Wrapper should provide a meaningful description of the source

•  $V1(a) \sqsubseteq \text{cites}(a,b), \text{cites}(b,a)$

# Query Containment: Containment Mappings

- $Q2 \subseteq Q1$  iff  $\forall$  databases  $D$ ,  $Q2(D) \subseteq Q1(D)$
- $Q2 \subseteq Q1$  iff there is a *containment mapping* from  $Q1$  to  $Q2$ :
  - mapping  $h$  of every atom in the body of  $Q1$  to an atom in the body of  $Q2$ ; and of the head variables of  $Q1$  to those in the head of  $Q2$



# Reviewing the Bucket algorithm

distinguished variable

existential variable

$V1(a) \rightarrow \text{cites}(a,b), \text{cites}(b,a)$

$V2(c,d) \rightarrow \text{sameTopic}(c,d)$

$V3(f,h) \rightarrow \text{cites}(f,g), \text{cites}(g,h), \text{sameTopic}(f,g)$

$V4(j) \rightarrow \text{sameTopic}(i,j)$

Q: Why don't we use V4?

A: Query needs 1<sup>st</sup> arg of sameTopic, but V4 does not output it.

User Query (using mediator relations):

$q(x) \leftarrow \text{cites}(x,y), \text{cites}(y,x), \text{sameTopic}(x,y)$

BUCKET:  $V1(x)$   $V1(x)$   $V2(x,y)$   
 $V3(x,h1)$   $V3(f1,x)$   $V3(x,y1)$

***Distinguished variables in the query must map to distinguished variables in the view!***

**(2x2x2=) 8 rewritings to check for containment: expensive!**

1<sup>st</sup> candidate:  $q'(x) \leftarrow V1(x), V1(x), V2(x,y)$

simplifies to:  $q'(x) \leftarrow V1(x), V2(x,y)$

is the unfolding  $q'(x) \leftarrow \text{cites}(x,y1), \text{cites}(y1,x), \text{sameTopic}(x,y)$  contained in  $q$ ?

# Insight: Certain variables/predicates need to be mapped together

- $q(x) \leftarrow \text{cites}(x,y), \text{cites}(y,x), \text{sameTopic}(x,y)$
- $V1(a) \rightarrow \text{cites}(a,b), \text{cites}(b,a)$ 
  - mapping  $\phi = \{x \rightarrow a, y \rightarrow b\}$
- To use  $V1$  we need to have access to 'b' so we can join it with  $\text{sameTopic}(a,b)$  (from some other source)
  - it is not because query asks for b but because query joins it

## Minicon's Property 1:

- If  $\phi(y)$  is an **existential** variable in the view, then  $\phi$  should “map” all query predicates that contain  $y$
- Intuitively, the view should cover the entire subset of the query predicates connected through  $y$

# Back to Example

**V1**(a)  $\rightarrow$  cites(a,b), cites(b,a)

**V2**(c,d)  $\rightarrow$  sameTopic(c,d)

**V3**(f,h)  $\rightarrow$  cites(f,g), cites(g,h), sameTopic(f,h)

**V4**(j)  $\rightarrow$  sameTopic(i,j)

User Query (using mediator relations):

$q(x) \leftarrow \text{cites}(x,y), \text{cites}(y,x), \text{sameTopic}(x,y)$

BUCKET:      **V1**(x)              **V1**(x)              **V2**(x,y)  
                 **V3**(x,h1)          **V3**(f1,x)          **V3**(x,y1)

**8 rewritings to check for containment: Containment expensive!**

Can we do better given the insight we just saw?

We shouldn't put V1 in the buckets, because it will fail!

To check this, instead of buckets of predicates, Minicon focuses on *variables*, using data structures called **MiniCon Descriptions (MCDs)**

# MiniCon: A “smarter” bucket algorithm

- In many cases, we don't need to perform the cross-product of all items in all buckets
- MiniCon concentrates on variables rather than atoms to create buckets called MiniCon Descriptions (MCDs)
- Combine MCDs that only overlap on distinguished view variables
- Eliminates the need for the containment check

# The MiniCon Algorithm (abstract)

- PHASE1: Form all MCDs that map all query variables that have to be mapped together
- PHASE2: Combine MCDs that only overlap on distinguished variables



# Minicon Descriptions (MCDs)

- A modification to the bucket:
  - “head homomorphism” – defines what variables must be equated
  - Mapping of variable names
  - Info about the covered query predicates

## Minicon's Property 1:

- If  $\varphi(y)$  is an ***existential*** variable in the view, then  $\varphi$  should “map” all query predicates that contain  $y$
- Intuitively, the view should cover the entire subset of the query predicates connected through  $y$

# MCDs for our example

- Form all MiniCon Descriptions (MCDs) that map all query variables that have to be mapped together

$q(x) \leftarrow \text{cites}(x,y), \text{cites}(y,x), \text{sameTopic}(x,y)$

$V1(a) \rightarrow \text{cites}(a,b), \text{cites}(b,a)$

$V2(c,d) \rightarrow \text{sameTopic}(c,d)$

$V3(f,h) \rightarrow \text{cites}(f,g), \text{cites}(g,h), \text{sameTopic}(f,g)$

$V4(j) \rightarrow \text{sameTopic}(i,j)$

## MCDs:

View	Mapping	Query Atoms Mapped
$V3(f,f)$	$x \rightarrow f, y \rightarrow g,$ $x \rightarrow h, f=h$	$\text{cites}(x,y), \text{cites}(y,x),$ $\text{sameTopic}(x,y)$ (atoms 1,2,3)
$V2(c,d)$	$x \rightarrow c, y \rightarrow d$	$\text{sameTopic}(x,y)$ (atom 3)

# MCD Construction

Form buckets (MCDs) more intelligently:

- Ask what is the minimal set of query subgoals that must be covered (via mappings) by each view
- First, look at join conditions in  $q$
- Then, follow joins on existential variables in views

For each atom of the query

For each atom of each view

Choose the least restrictive head homomorphism to match the atom of the query

If we can find a way of mapping the variables, then add MCD for each possible “maximal” extension of the mapping that satisfies Property 1

# General rules for mapping variables

- For a query subgoal  $G$  and a view subgoal  $H$  in view  $V$ , the MiniCon algorithm considers a mapping from  $G$  to  $H$
- In this mapping, a query variable  $X$  is mapped to a view variable  $A$
- Four possible cases:
  - Case 1:  $X$  is dist.,  $A$  is dist.. OK.
    - $A$  is exported, so can join with other views.
  - Case 2:  $X$  is exist.,  $A$  is dist.. OK.
    - Same as above
  - Case 3:  $X$  is dist.,  $A$  is nondist.. NOT OK.
    - $X$  needs to be in the answer, but  $A$  is not exported.
  - Case 4:  $X$  is nondist.,  $A$  is nondist..
    - Then all the query subgoals using  $X$  must be able to be mapped to subgoals in view  $V$ .
    - Reason: since  $A$  is not exported in  $V$ , it's impossible for  $V$  to join with other views to answer conditions involving  $X$ .
    - I.e., “either NONE or ALL.”

# MCD Formation for our example

Consider  $V1(a) \rightarrow \text{cites}(a,b), \text{cites}(b,a)$

- Can cover query subgoal  $\text{cites}(X,Y)$
- To do this, we map:  $x \rightarrow a, y \rightarrow b$
- $y$  is an existential join variable
- So  $V1$  needs to cover the rest of the query atoms that contain  $y$ , which are:

$\text{cites}(y,x)$  and  $\text{sameTopic}(x,y)$

- $V1$  can cover the  $\text{cites}$  subgoal ( $x \rightarrow a, y \rightarrow b$ ), but not the  $\text{sameTopic}$
- Hence, no MCD is created for  $V1$

# MCD Formation for our example

Consider  $V2(c,d) \rightarrow \text{sameTopic}(c,d)$

- Can cover  $\text{sameTopic}(x,y)$  (and nothing else)
- To do this, we map:  $x \rightarrow c, y \rightarrow d$
- The MCD says how the query atoms may be covered by  $V2$

## MiniCon Descriptions (MCDs)

View	Mappings	Query Atoms Covered
V2	$x \rightarrow c, y \rightarrow d$	3

# MCD Formation for our example

Consider  $V3(f,h) \rightarrow \text{cites}(f,g), \text{cites}(g,h), \text{sameTopic}(f,g)$

- Can cover query subgoal  $\text{sameTopic}(x,y)$
- To do this, we map:  $x \rightarrow f, y \rightarrow g$
- $y$  is an existential join variable
- So  $V3$  needs to cover the rest of the query atoms that contain  $Y$ , which are:  
 $\text{cites}(x,y)$  and  $\text{cites}(y,x)$
- To do this, we map:  $x \rightarrow f, y \rightarrow g$  and  $x \rightarrow h$

# MCD Formation for our example

We get the following MCD for

**V3**(f,h)  $\rightarrow$  cites(f,g), cites(g,h), sameTopic(f,g)

**MiniCon Descriptions (MCDs)**

View	Mappings	Query Atoms Covered
V2	$x \rightarrow c, y \rightarrow d$	3
V3	$x \rightarrow f, y \rightarrow g, x \rightarrow h$	1,2,3

- **x** is mapped to both **f** and **h**
- Can we fix this?
- **Yes**, if both **f** and **h** are **distinguished** in **V3**
- **No**, otherwise



# MCD Formation for our example

We fix the problem by making **f** and **h equal** when producing the rewriting:

Query rewriting:  $q'(f) :- V3(f,f)$

Since **V3** covers all query atoms, no other view is needed

# Phase 2: MCD Combination

- Now look for all ways of combining **pairwise disjoint** subsets of MCDs, covering the **entire** query
  - Greatly reduces the number of candidates!
  - Also proven to be correct without the use of a containment check

## MCDs:

View	Mapping	Query Atoms Covered
V3(f,f)	$x \rightarrow f, y \rightarrow g,$ $x \rightarrow h, f=h$	cites(x,y), cites(y,x), sameTopic(x,y)
V2(c,d)	$x \rightarrow c, y \rightarrow d$	sameTopic(x,y)

## Rewriting:

$$q'(x) \leftarrow V3(x,x)$$

$q(x) \leftarrow \text{cites}(x,y), \text{cites}(y,x), \text{sameTopic}(x,y)$

**V1**(a)  $\rightarrow \text{cites}(a,b), \text{cites}(b,a)$

**V2**(c,d)  $\rightarrow \text{sameTopic}(c,d)$

**V3**(f,h)  $\rightarrow \text{cites}(f,g), \text{cites}(g,h), \text{sameTopic}(f,g)$

**V4**(j)  $\rightarrow \text{sameTopic}(i,j)$

# MiniCon Example 2

Query:

$q(X) \text{ :- cites}(X,Y), \text{ cites}(Z,X), \text{ inSIGMOD}(X)$

Views:

$V7(A) \text{ :- cites}(A,B), \text{ inSIGMOD}(A)$

$V8(C) \text{ :- cites}(D,C), \text{ inSIGMOD}(C)$

## MiniCon Descriptions (MCDs)

Step 1:

View	Mappings	Query Atoms Covered
V7	$X \rightarrow A, Y \rightarrow B$	1
V7	$X \rightarrow A$	3
V8	$Z \rightarrow D, X \rightarrow C$	2
V8	$X \rightarrow C$	3

# MiniCon Example 2

## MiniCon Descriptions (MCDs)

View	Mappings	Query AtomsCovered
V7	$X \rightarrow A, Y \rightarrow B$	1
V7	$X \rightarrow A$	3
V8	$Z \rightarrow D, X \rightarrow C$	2
V8	$X \rightarrow C$	3

Step 2:

Query rewriting 1:  $q_1(X) :- V7(X), V8(X), V7(X)$

Query rewriting 2:  $q_2(X) :- V7(X), V8(X), V8(X)$

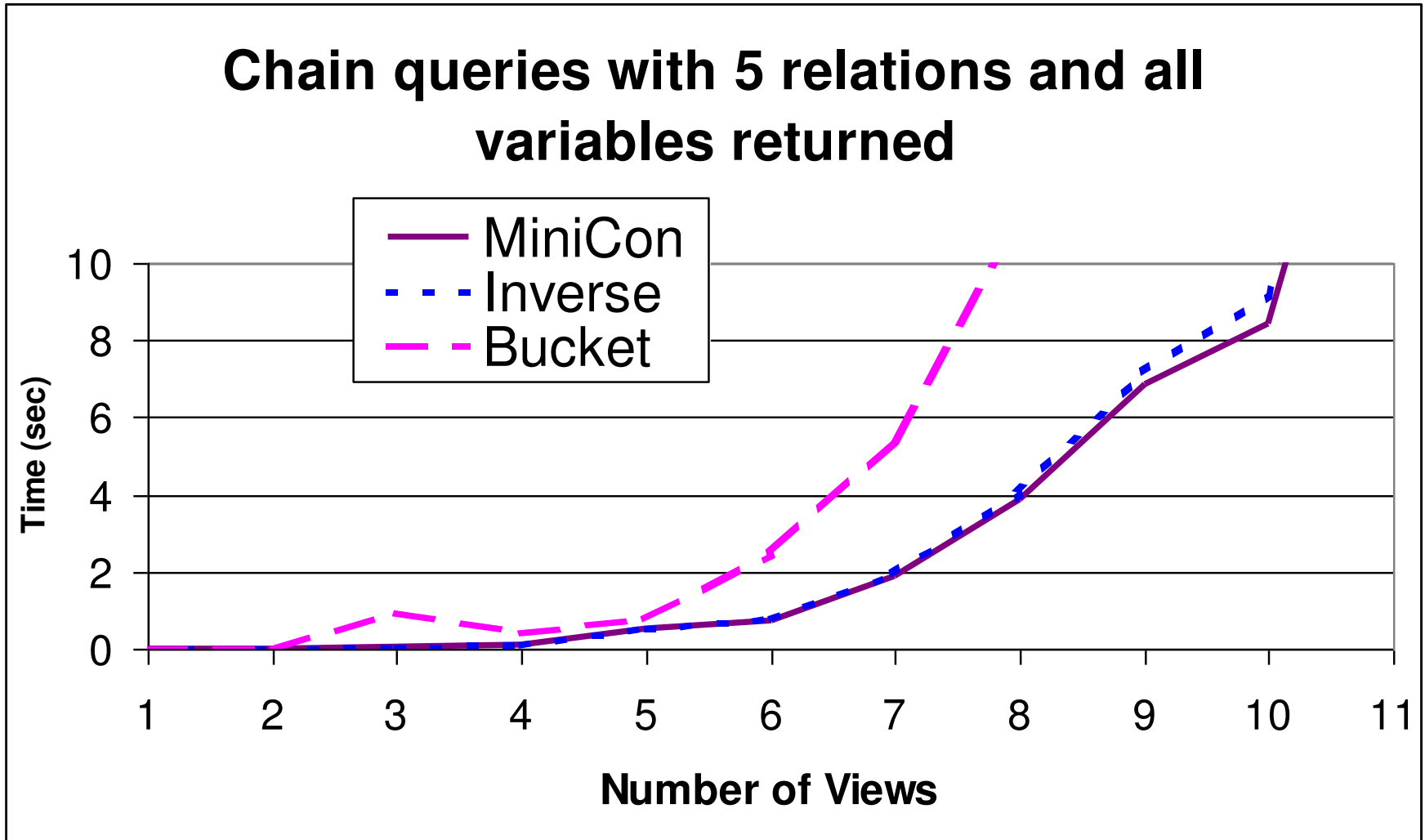
Final rewriting:  $q'(X) :- V7(X), V8(X)$

# The MiniCon Algorithm

- PHASE1: Form all MCDs that map all query variables that have to be mapped together
  - For all query predicates, all view predicates
    - Map query predicate to view (find a covering)
    - Map whole (existentially) connected subset of the query when covering a join variable existentially (Property 1)
      - Subset should be minimal (Property 2)
- PHASE2: Combine MCDs that only overlap on distinguished variables
  - **Choose** over *mutually exclusive* MCDs where their union covers the entire query
  - Generate conjunctive rewritings
  - “Minimize” rewritings

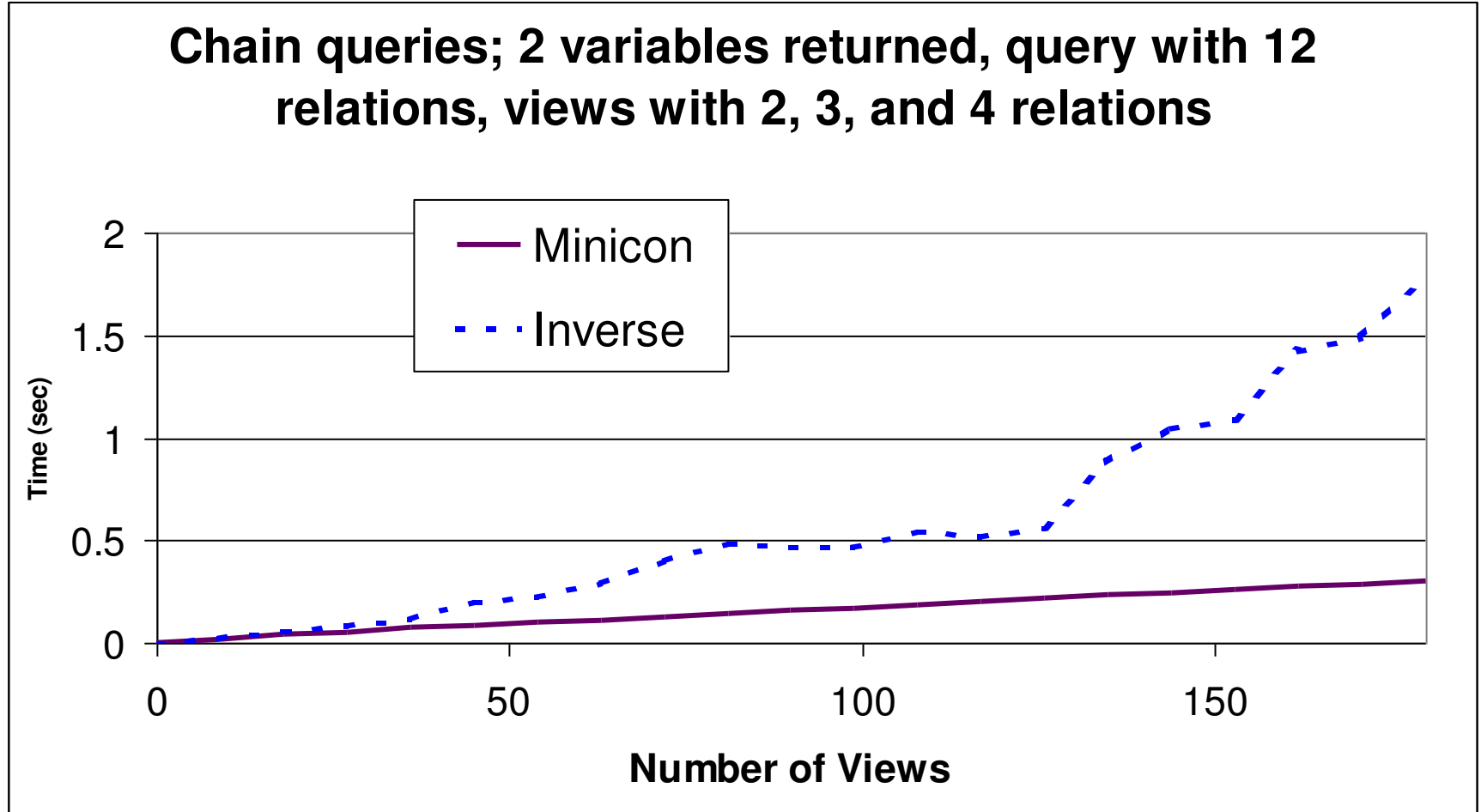
# Experimental results:

## Many rewritings



# Experimental Results:

## Few rewritings

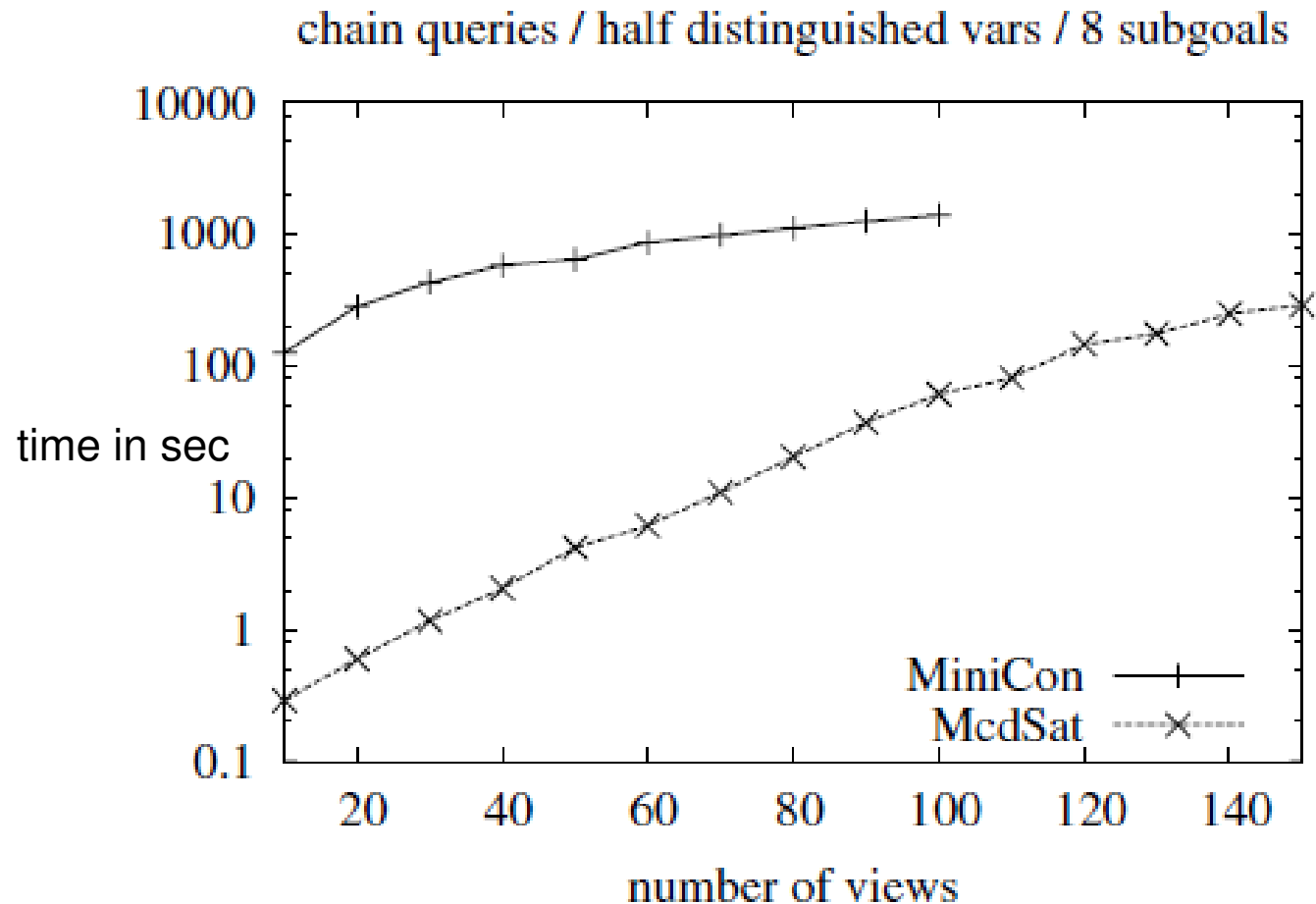


# MCDSAT

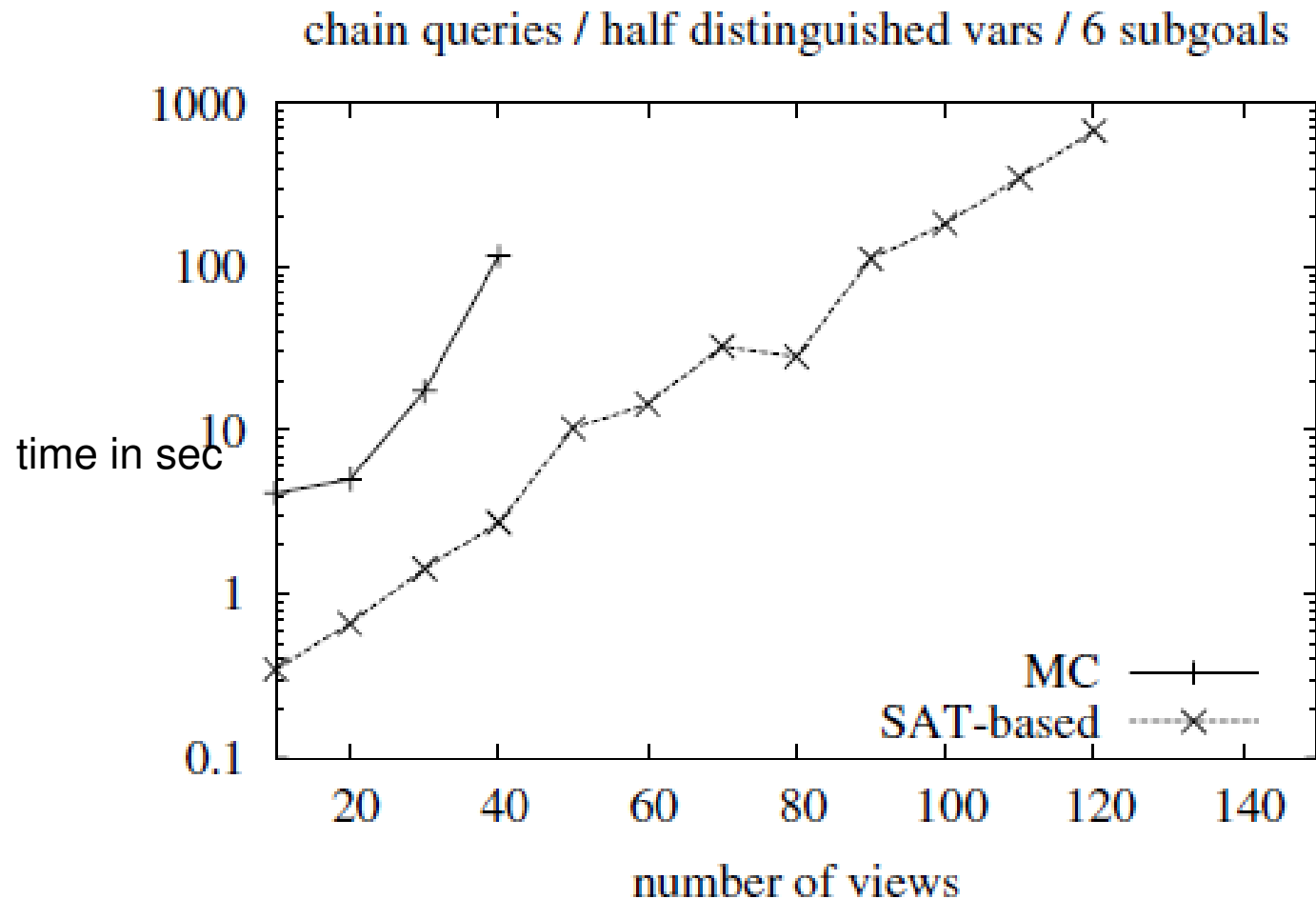
- Cast MiniCon a SAT problem
  - Cast minicon's first phase (the MCDs generation problem) into a propositional theory whose models constitute the MCDs.
    - Minicon's mapping and properties 1 and 2 result in clauses of that theory, which is further extended with more optimization clauses.
  - Compile theory into normal form d-DNNF
    - Expensive to compile!
  - d-DNNF implements model enumeration in polynomial time
- For the second phase, that of MCD combination and rewriting generation, MCDSAT considers
  - either a traditional implementation of MiniCon's phase2 or
  - additional clauses are devised in order to present an extended logical theory whose models are in correspondence with the rewritings



# Experiments: MCDs generation



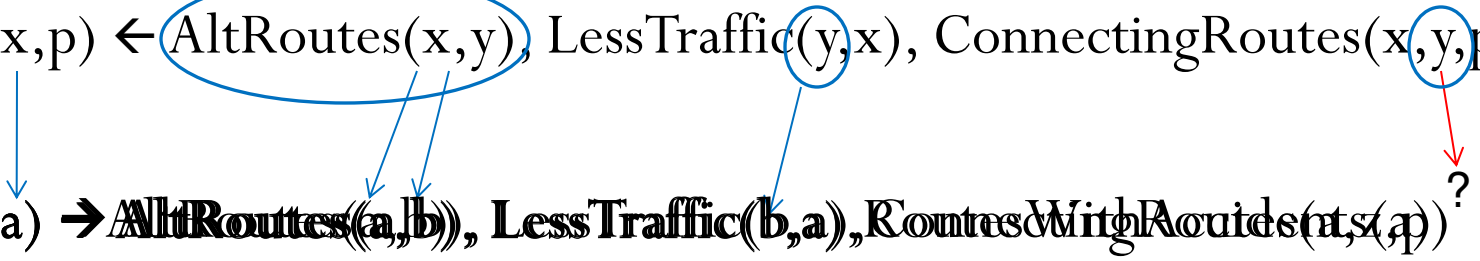

# Experiments: Rewriting generation



# GQR Approach

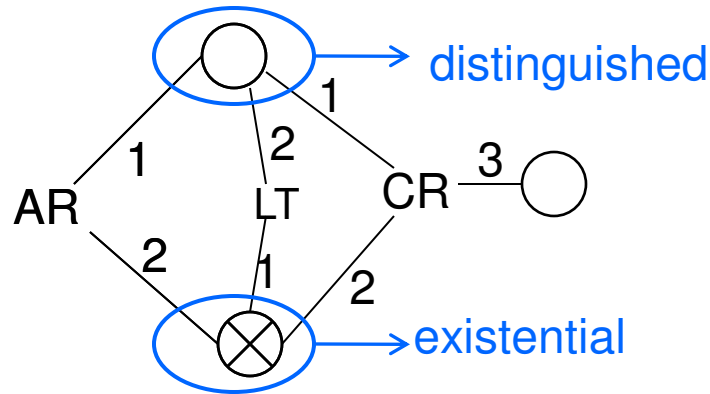
- Process sources offline
  - Find common patterns of predicates
    - Represent each pattern just once
    - Index patterns
  - Record join information
- Map query against this compact, indexed representation of sources
- Build rewriting incrementally
- Map the joins respecting the existential/distinguished variable requirements
- Efficient reformulation
- Fail-fast conditions
- Scalable as the number of sources grows

# Saving redundant work

- MiniCon and MCDSAT do redundant work
- $Q(x,p) \leftarrow \text{AltRoutes}(x,y), \text{LessTraffic}(y,x), \text{ConnectingRoutes}(x,y,p)$ 

- $S_g(a) \rightarrow \text{AltRoutes}(a,b), \text{LessTraffic}(b,a), \text{RoutesWithAcids}(a,b,p)$ 

- Idea: *compactly represent the different patterns across sources by using just one graph for the same pattern wherever this appears!*
  - This can even be done offline (a priori to any query)
- Using graphs we can abstract from the variable names
  - only variable *type* and *position* are needed to decide on a covering

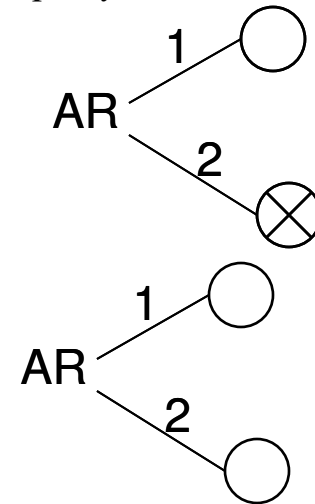
# Our Modeling: Query/View Graphs

- $Q(x,p) \leftarrow \text{AltRoutes}(x,y), \text{LessTraffic}(y,x), \text{ConnectingRoutes}(x,y,p)$



Predicate Join Pattern (PJ)

Potential View PJs that can cover query PJ for AR

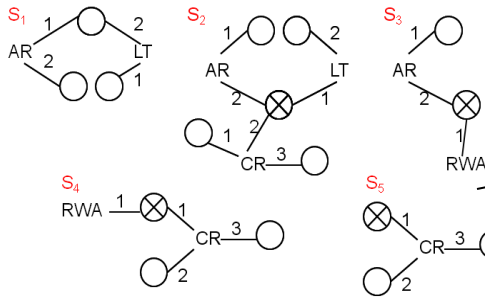


- Our approach: Map subgraphs of the query to subgraphs of sources
  - Smallest subgraphs we consider: PJs

# Our Approach

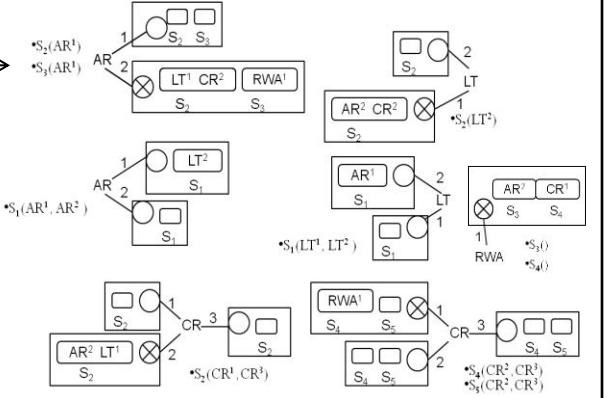
## Source Preprocessing

### Sources



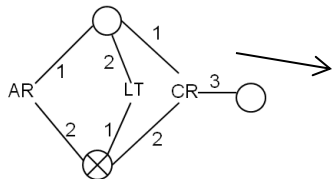
- Source Decomposition,
- Unique representation of common patterns into PJs,
- Bookkeeping

### Indexing



## Query reformulation

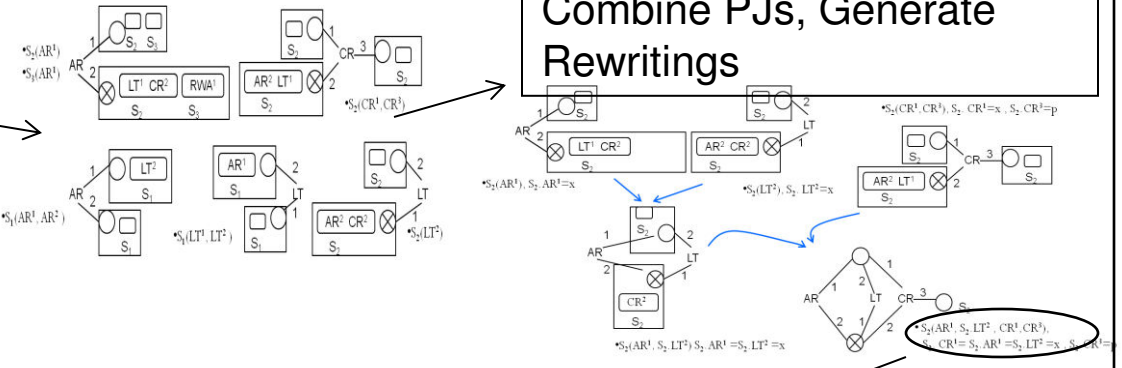
### Query



$Q(x,p) :- AR(x,y),$   
 $LT(y,x), CR(x,y,p)$

Retrieve  
relevant  
source  
PJs

Combine PJs, Generate  
Rewritings



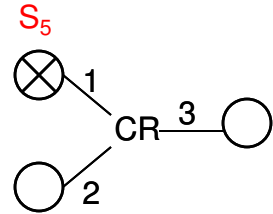
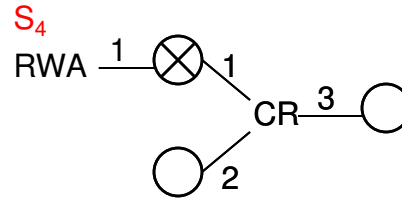
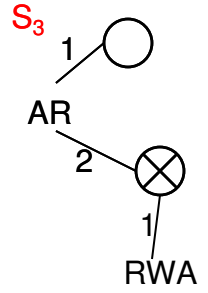
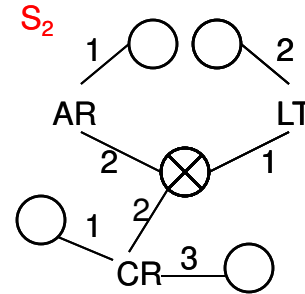
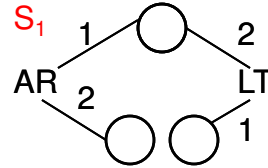
Rewriting:  $Q(x,p) :- S_2(x,x,x,p)$

# Source Preprocessing:

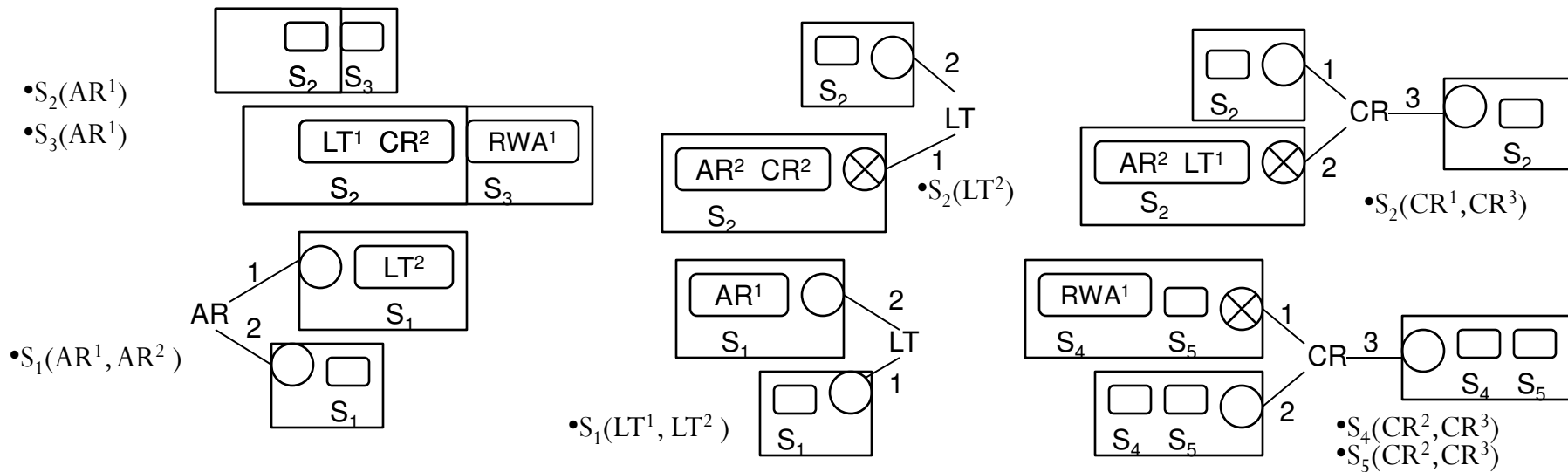
## Creating graphs, bookkeeping, indexing

- Sources

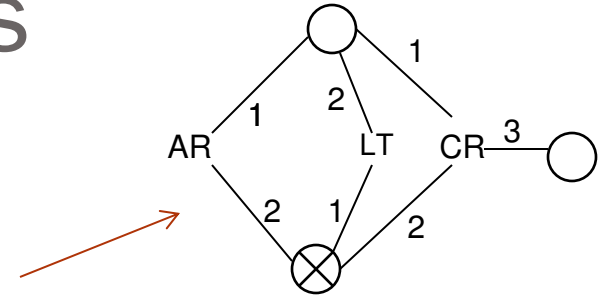
- $S_1(a,b,c) \rightarrow AR(a,b), LT(c,a)$
- $S_2(d,f,m,q_1) \rightarrow AR(d,e), LT(e,f), CR(m,e,q_1)$
- $S_3(g) \rightarrow AR(g,h), RWA(h)$
- $S_4(j,p_1) \rightarrow CR(i,j,p_1), RWA(i)$
- $S_5(l,p_2) \rightarrow CR(k,l,p_2)$



- Data Structures: **shared** patterns, joins



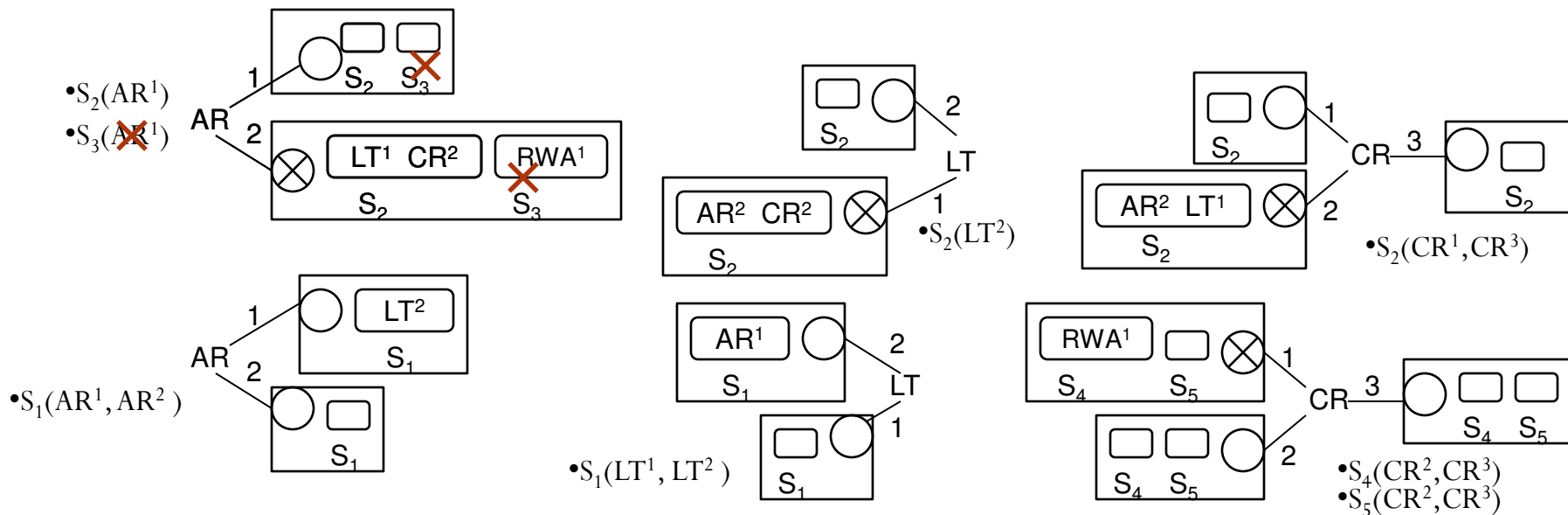
# Finding relevant sources for user query



- Given user query:

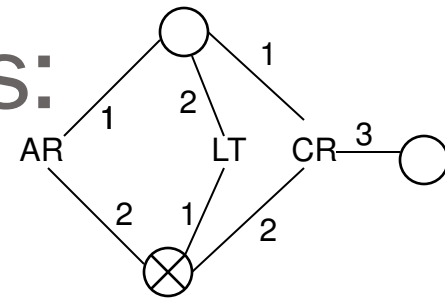
$Q(x,p) \leftarrow \text{AltRoutes}(x,y), \text{ LessTraffic}(y,x), \text{ ConnectingRoutes}(x,y,p)$

- Retrieve source PJs that *cover* the query PJs, then
  - drop irrelevant sources, **prune early**
  - drop non-usable PJs
  - Fail early** if no valid source PJ, query unsatisfiable

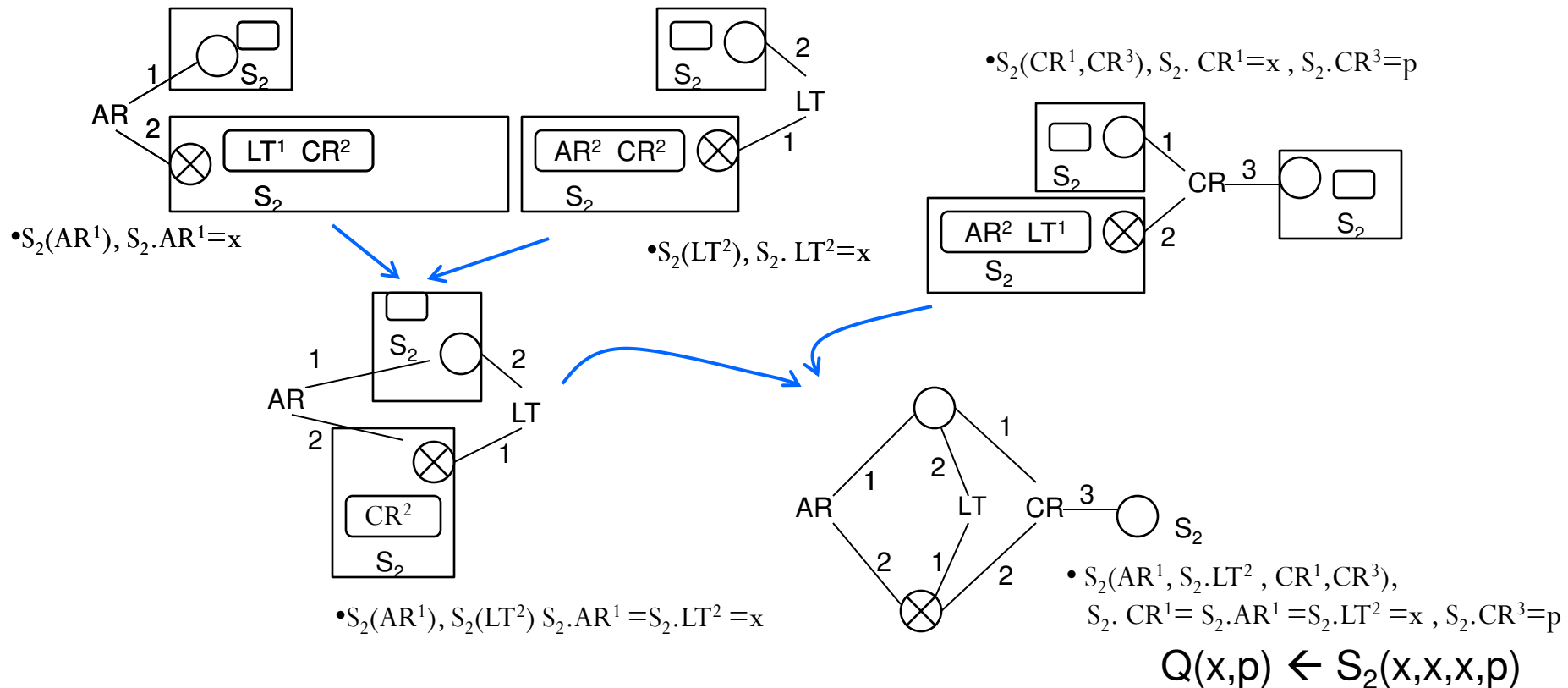




# Building the query rewritings: Combine graphs (PJs)

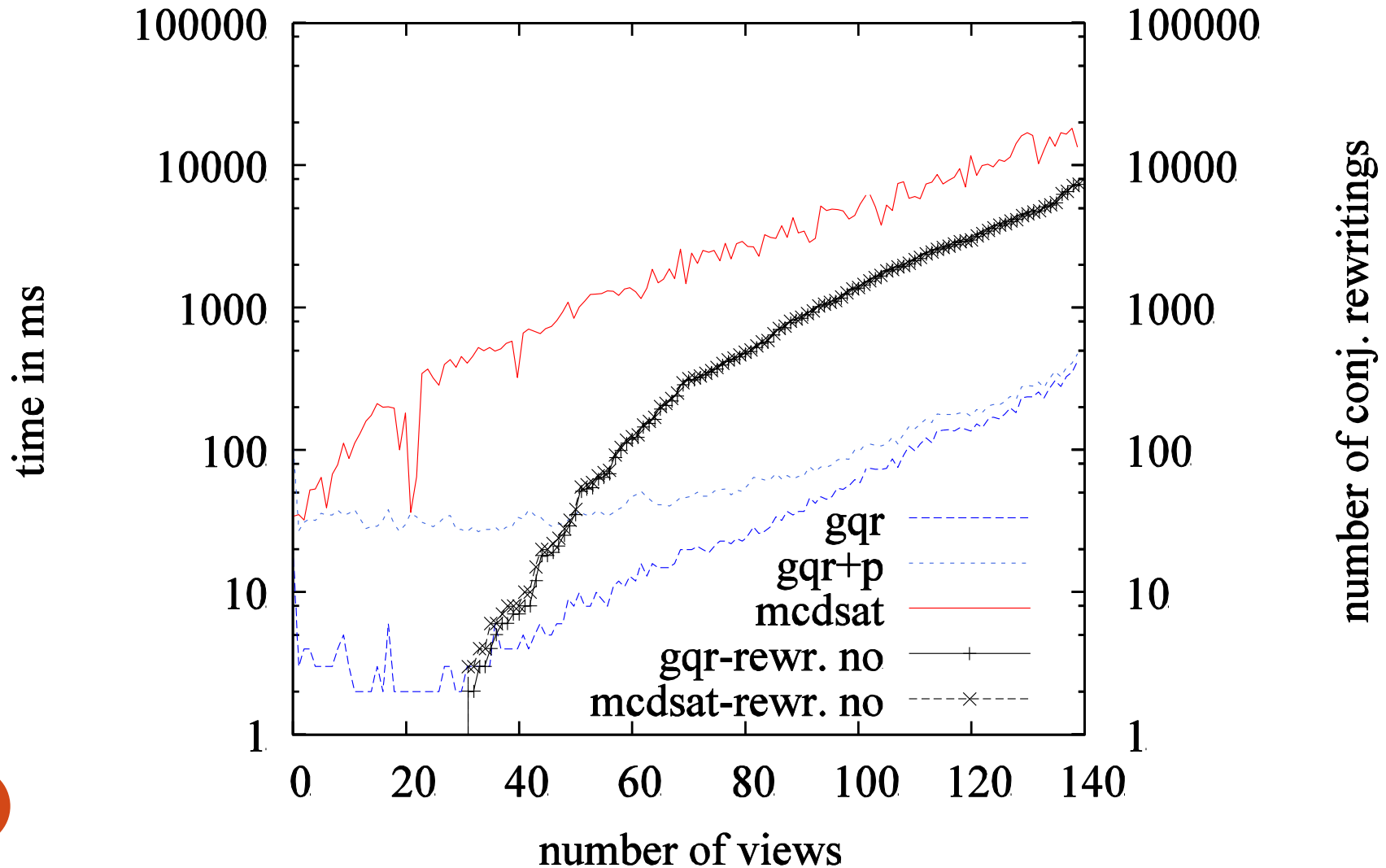


- Continuously combine source PJs to a larger ones until we cover the entire query or fail
  - Can only combine same type nodes
  - Fails to combine if existential joins are not covered
  - As you combine you build the rewriting



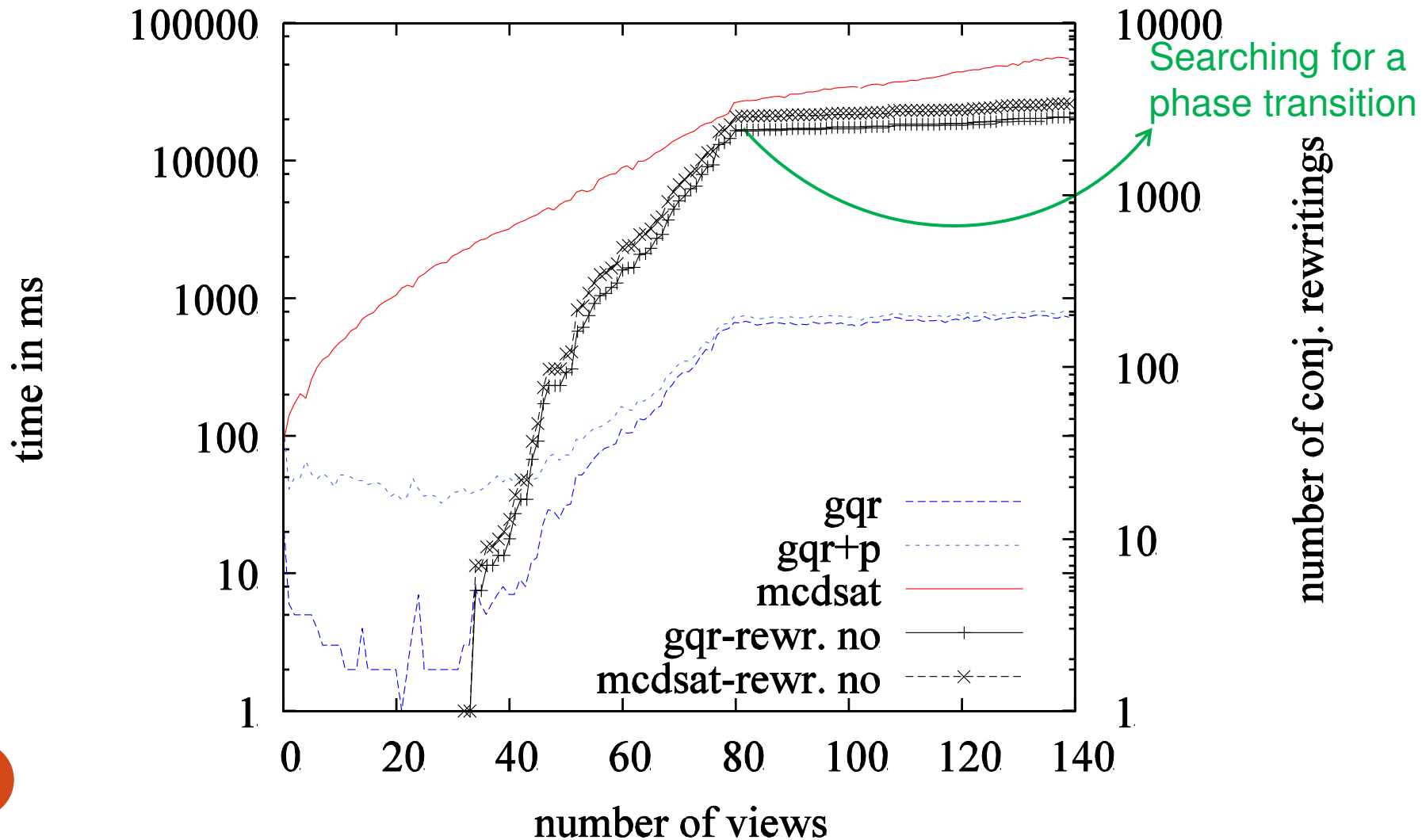
# Experimental Evaluation: Comparison with MCDSAT

99 star queries, 140 star views/q, 8 preds, 5 preds/body,  
4 vars/pred, 10 dist vars,  $\leq 5$  rep. preds/query

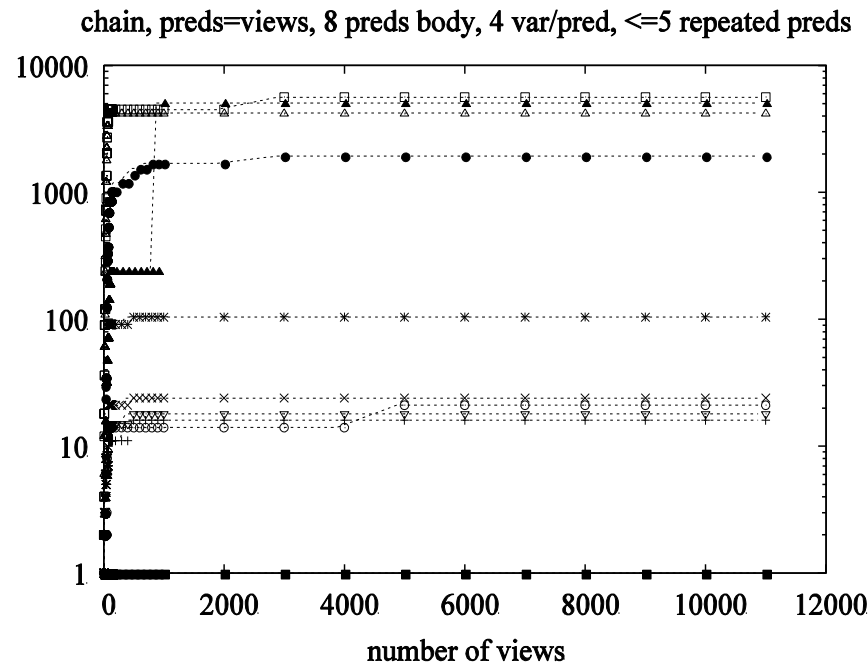
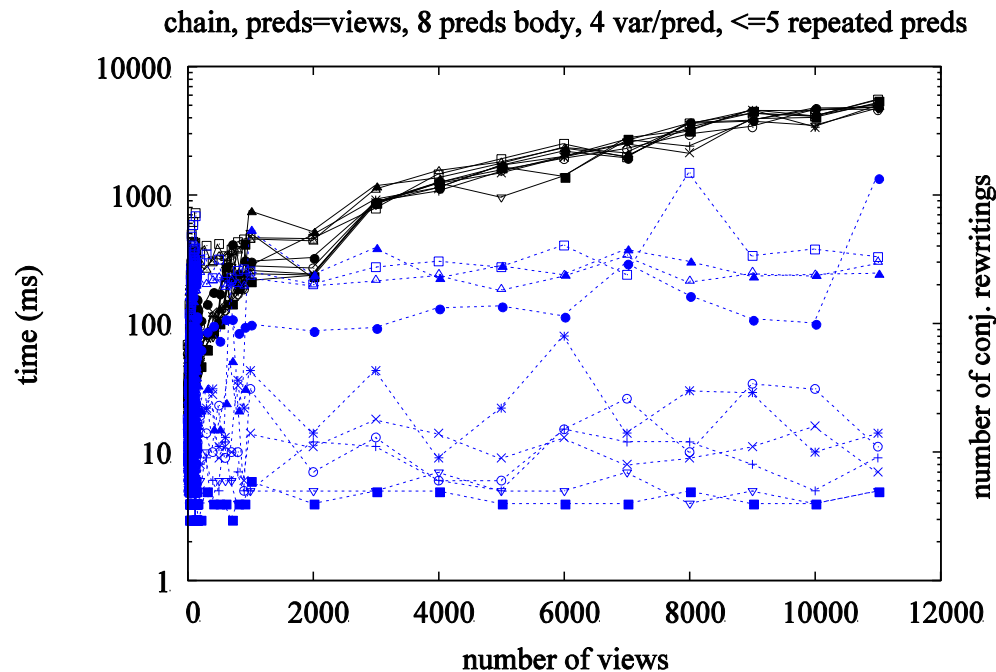


# Experimental Evaluation: Comparison with MCDsAI

93 chain queries, 140 chain views/q, 20 preds, 8 preds/body, 4 vars/pred,  
10 dist vars/1-80 views, 3 dist vars/80-140 views,  $\leq 5$  rep. preds/q



# Experimental Evaluation: Scaling to 10000 views



Pushing the exponential computation offline  
GQR Time proportional to number of rewritings

# Why does GQR perform better?

- Preprocessing = Step1 of minicon with an additional burden
  - Process ALL subgoals of ALL views
    - Construct PJ patterns
  - Construct infoboxes
  - + indexing = more optimization (cost of indexing can be amortized)
- Then we do something ``lighter than'' step 2 of Phase 1
  - Combine CPJs
  - And in less iterations! (No. of atomic query subgoal X No. of PJs)
- MiniCon's Phase 2 is *entirely encapsulated* in our combination of CPJs.

# GQR Discussion

- Contributions
  - Compact graph representation of views
    - Combine/reject whole set of views/rewritings at once
  - Fail-fast conditions
  - Scalable reformulation
    - Even more efficient by pushing computation offline
- See paper for:
  - Repeated predicates
  - Minimal rewritings and minimization conditions
  - Index on the preprocessed PJs
  - Detail comparison with Related Work (Minicon, MCDSAT, Hypergraph decompositions)

# GQR playground (open problems)

- Paying more for indexing during preprocessing will make the retrieval of source patterns cheap – Explore tradeoff
- Minimal rewritings will yield an optimized query
- Repeated predicates are a big deal
- Order of combination of patterns could have an impact
- Future work:
  - adding constants and built-in predicates
  - more source preprocessing
  - UCQs as inputs
  - richer mediator languages (e.g., description logics)

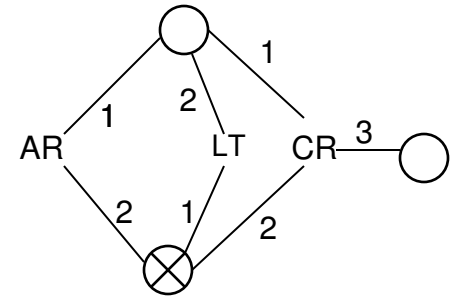
QUESTIONS?

*[konstant@usc.edu](mailto:konstant@usc.edu)*

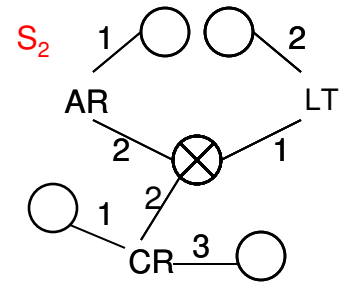


# Example

- $Q(x,p) :- \text{AltRoutes}(x,y), \text{LessTraffic}(y,x), \text{ConnectingRoutes}(x,y,p)$



- $S_2(d,f,m,q_1) :- \text{AltRoutes}(d,e), \text{LessTraffic}(e,f), \text{ConnectingRoutes}(m,e,q_1)$



- Rewriting:  $Q(x,p) :- S_2(x,x,x,p)$

# Experimental Evaluation: Searching for a phase transition

Try to scale to 10000 sources

Exponential number of rewritings

- run out of memory at in the 250K-1M rewritings

Fail fast

Search for a phase transition

