

Efficient Semantic Querying of Relational Databases with Resolution

Alexandre Riazanov

RuleML Initiative

CSWWS, May 24, 2009

Semantic Querying of RDB

Main scenario:

- One or more relational databases (RDB). Tables are conceptually treated as sets (conjunctions) of ground logical assertions:

UNIV_DB_TAKES_COURSE	STUDENT	COURSE	...
	s1	c1	UNIV_DB_TAKES_COURSE(s1,c1)
	s2	c2	UNIV_DB_TAKES_COURSE(s2,c2)

- Knowledge bases for the domains: rule sets (Derivational RuleML) and taxonomies/ontologies (RDFS/OWL):

$$graduateStudent(X) : \neg takesCourse(X, C), graduateCourse(C)$$

- Semantic mapping for RDB schemas:

$$takesCourse(X, C) : \neg univ_db_takes_course(X, C)$$

- User formulates logical queries that have to be answered modulo the KB:

$$? - graduateStudent(S), memberOf(S, D), suborganization(D, 'UBC').$$

$$S = 'JohnSmith', D = 'PsychCS' ;$$

$$S = 'MaryTaylor', D = 'MathStatPhys' ;$$

...

Main Applications of Semantic Querying

Non-programmer interface to RDB:

- financial analysts, biomedical researchers, criminal investigators, patent examiners, etc., etc., don't (want to) know RDB programming!
But they are usually able to formulate *complex queries in terms of their domains*.
- a step towards querying in NL: logic is closer to English than SQL
- hundreds of thousands of potential end users

Web-scale semantic search:

- precise answers to queries; answers justified by reasoning with ontology axioms and rules
- probably the main prize in the game: millions of potential end users
- semantic querying of RDB may be a large step in this direction

Outline of the Proposed Technique

- Use a variant of the classical resolution calculus to answer queries.
This allows to use *very expressive knowledge bases and query languages*.
- Use resolution in a smart way: *find answers in bulk*, i.e., find schematic answers covering *whole sets* of concrete solutions.
- Convert schematic answers into SQL to leverage the efficiency of highly optimised RDBMS.
- The technique can be characterised as *incremental query rewriting* – single query can give rise to a potentially infinite set of SQL queries whose union covers all answers.

Outline of the Proposed Technique

- Use a variant of the classical resolution calculus to answer queries.
This allows to use *very expressive knowledge bases and query languages*.
- Use resolution in a smart way: *find answers in bulk*, i.e., find schematic answers covering *whole sets* of concrete solutions.
- Convert schematic answers into SQL to leverage the efficiency of highly optimised RDBMS.
- The technique can be characterised as *incremental query rewriting* – single query can give rise to a potentially infinite set of SQL queries whose union covers all answers.

Very expressive querying at reasonable cost.

Resolution

Main principle:

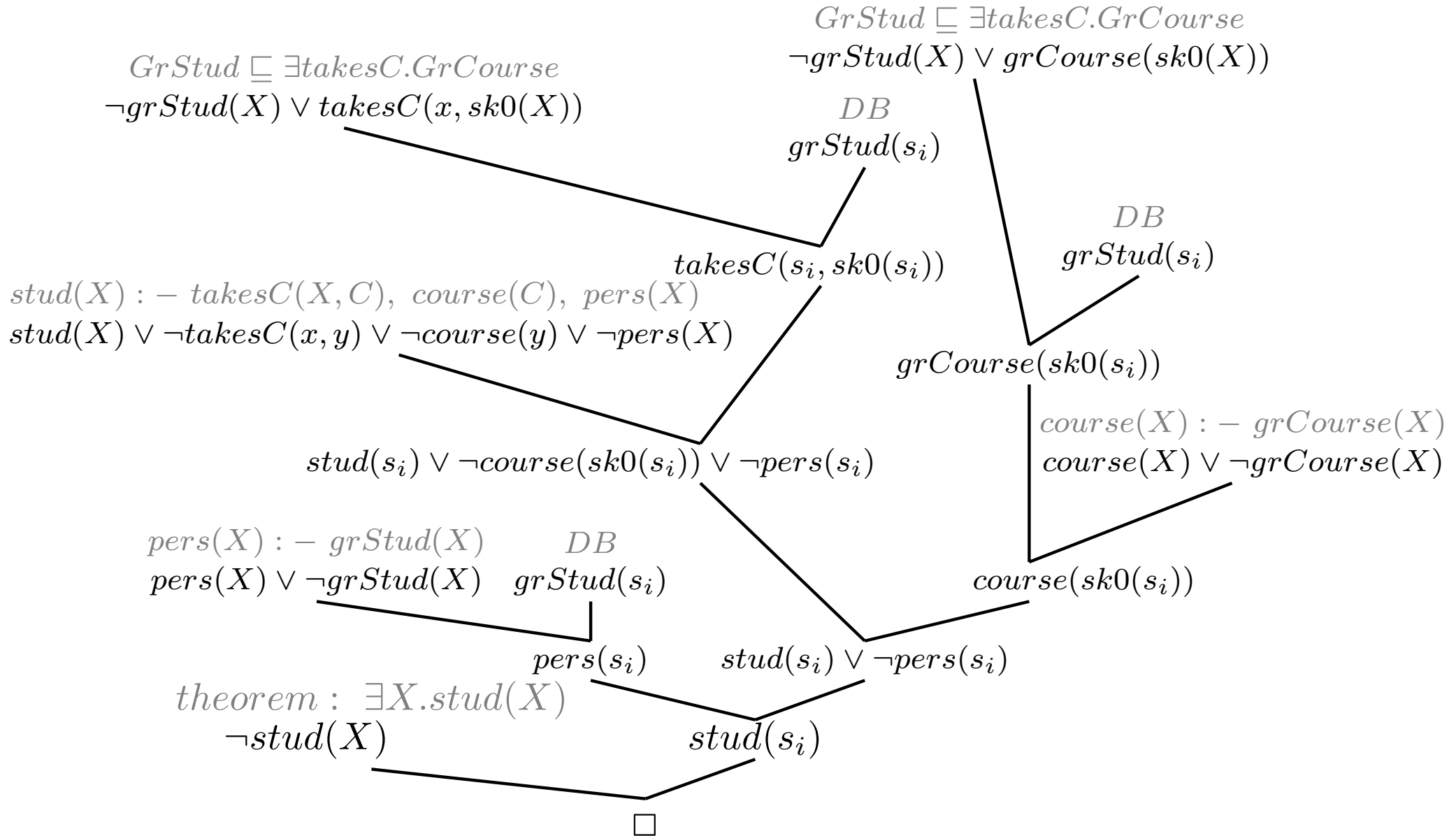
$$\begin{array}{ccc} C \vee A & & \neg B \vee D \\ & \searrow & \swarrow \\ & (C \vee D)\theta & \end{array} \quad \text{where } \theta = mgu(A, B)$$

Procedure: Given a set of clauses, exhaustively apply rules until an empty clause (contradiction) is found, in which case the original set is inconsistent.

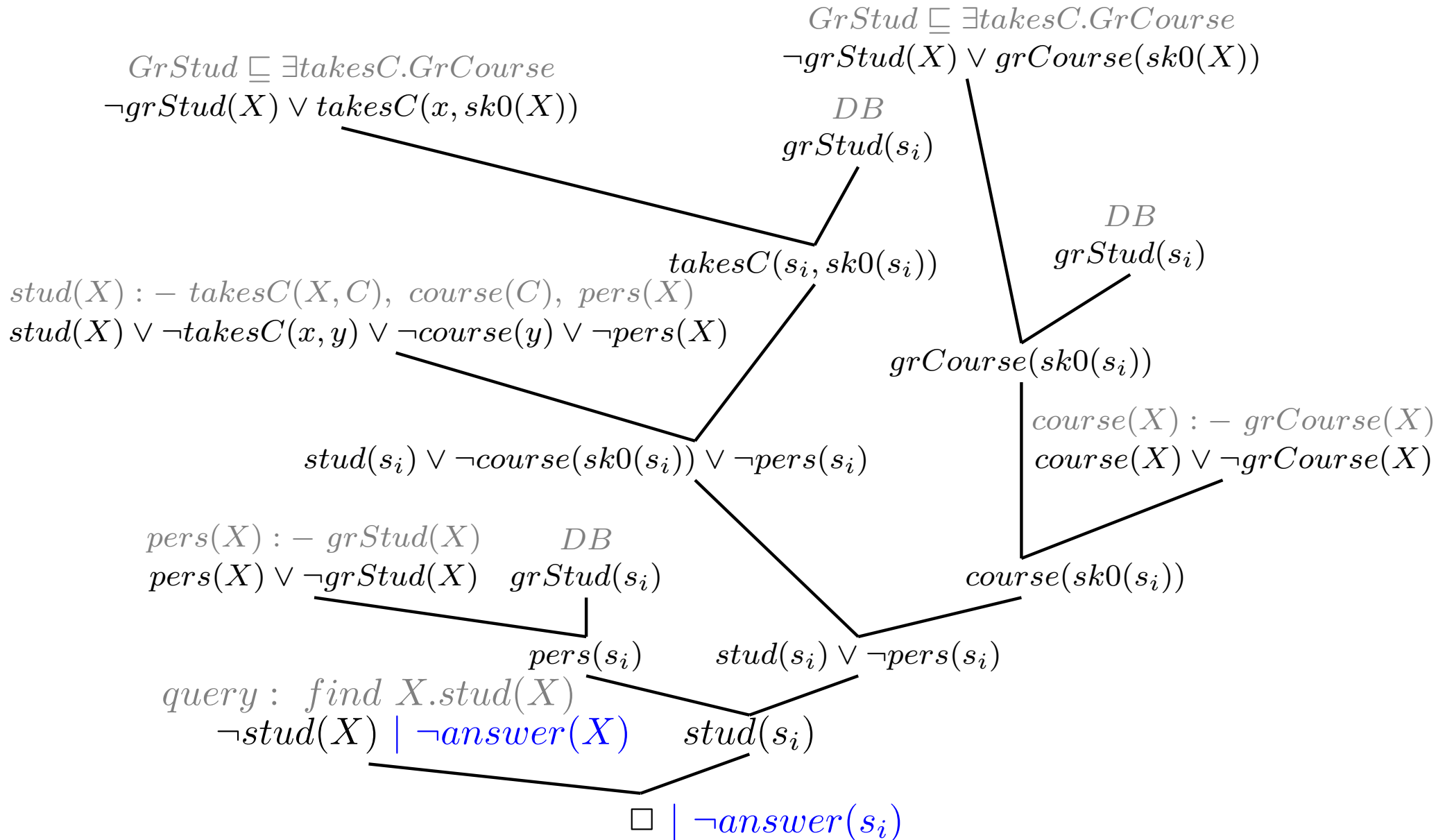
Well studied: powerful and elegant theory, > 40 years of research.

Efficient implementation techniques exist: datastructures, heuristic-driven algorithms, term indexing techniques, etc. Efficient (general purpose) implementations: E, Gandalf, Otter, SPASS, Vampire,... Ontology oriented optimisations: chain resolution.

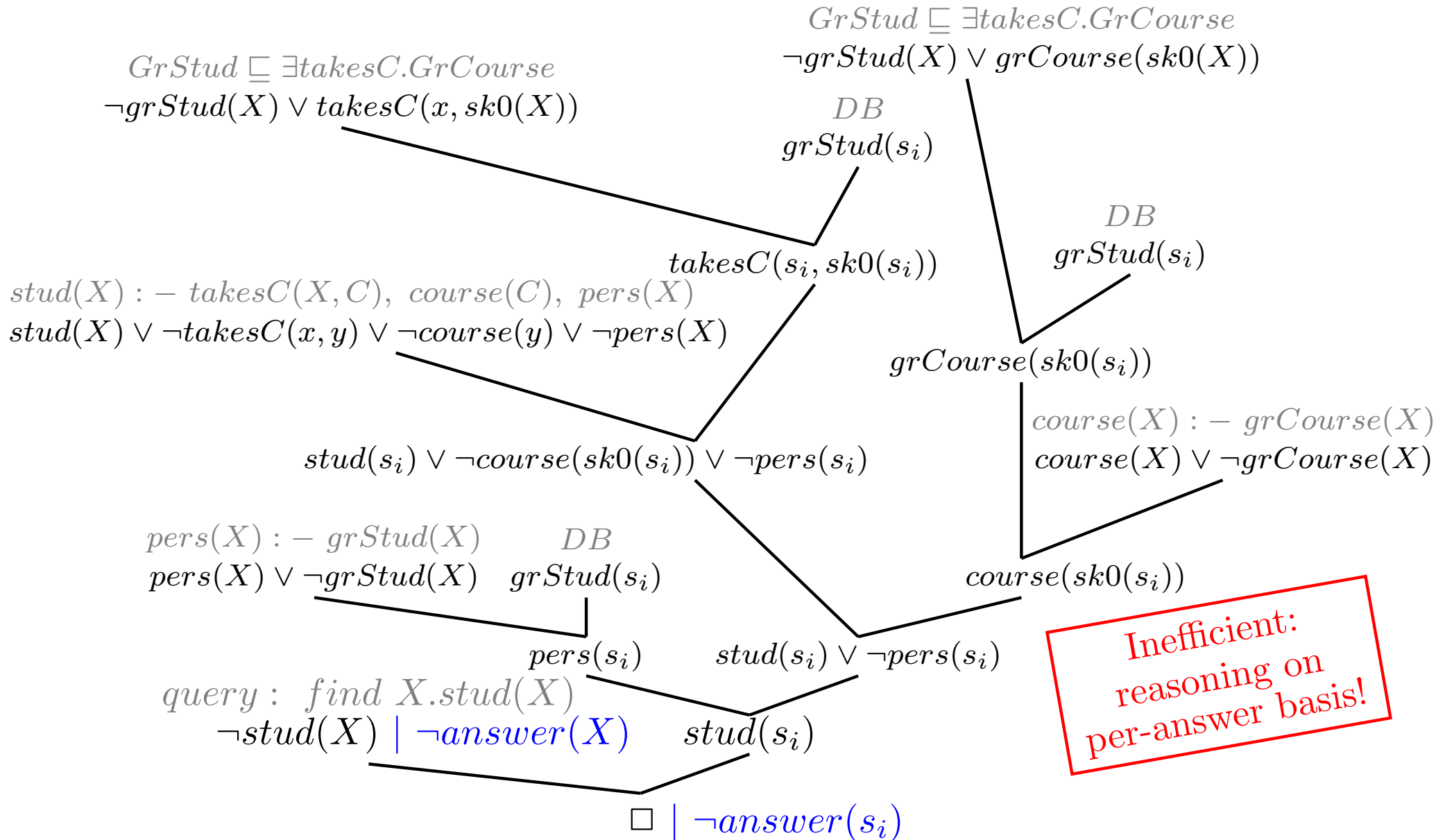
Query Answering with Resolution



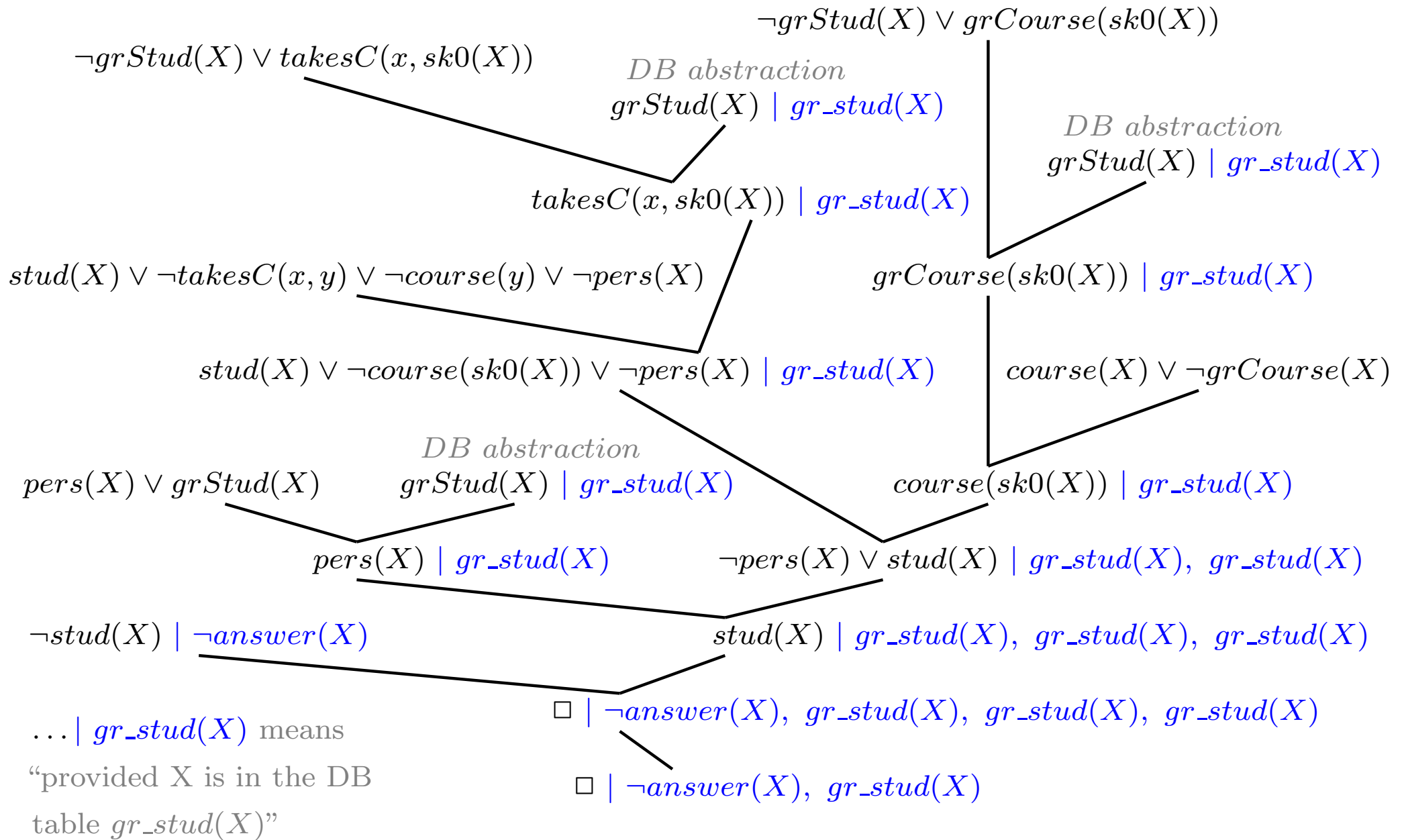
Query Answering with Resolution



Query Answering with Resolution



Schematic Answer Derivation



Transformation to SQL

Query: ? – $stud(S), memberOf(S, "http://www.ubc.ca")$.

Schematic answer:

□ | $\neg answer(S),$
 $takes_course(S, C),$
 $gr_course(C),$
 $member_of(S, D),$
 $suborganization(D, "http://www.ubc.ca")$

SQL query for this schematic answer:

```
SELECT takes_course.subject AS S
FROM takes_course, gr_course, member_of, suborganization
WHERE takes_course.object = gr_course.instance
AND takes_course.subject = member_of.subject
AND member_of.object = suborganization.subject
AND suborganization.object = "http://www.ubc.ca"
```

Soundness and Completeness w.r.t. Answers

- The method is sound: schematic answers cover only legitimate concrete answers, as long as the underlying resolution calculus is sound.
- Completeness concept: for every existing concrete answer, we must be able to find a schematic answer covering it.
- At the calculus level: *complete* with reasonable requirements on the calculus, if resolution or hyperresolution is used (details in the paper).
- Complete with subsumption? Intuitively OK, requires careful formalisation.
- Complete superposition-based calculus for equality? Completeness proof seems possible, requires careful research (soundness already established).

Work in Progress

- Prototype exists: the Vampire reasoner extended for query answering over DB abstractions + simple SQL generator. Currently supports TPTP, OWL, Derby and MySQL. Next target – monotonic Derivational RuleML (up to **foloqeq**).
- Some experiments have been done, mostly with LUBM: all queries are answered on a large instance with an unoptimised instance store. Next step: case study with a real-life RDB design and RuleML.
- A lot more theoretical work is possible: completeness of superposition for KBs with equality, completeness with standard search-space reduction techniques, termination.

Semantic Web Indexing

- Indexing problem: avoid downloading SW documents that are not (known to be) relevant to the query. Keyword-based indexing does not work: relevant data may have no URIs/data literals in common with the query.

Query: ? – *animal(X), ofBrightColour(X)*.

Relevant document: *elephant(jumbo). pink(jumbo)*.

- Solution idea: index documents by their abstractions. Download only those documents whose abstractions contribute to a schematic answer.

Indexes: $I1 = \textit{elephant}(X) \mid \textit{elephant}(X)$, $I2 = \textit{pink}(X) \mid \textit{pink}(X)$

One of the schematic answers: $\square \mid \neg\textit{answer}(X), \textit{elephant}(X), \textit{pink}(X)$,
derived from $I1$ and $I2$.

\Rightarrow download all documents indexed with $I1$ or $I2$.

- Just a conceptual framework, many optimisations are possible.

Related Work

Protégé-related project (Stanford Medical Informatics). Querying biomedical databases with OWL and SWRL. Using SQL to extract potentially relevant data. Using a rule engine to do the inference.

XSTONE project (Tallinn U. of Tech.) Very expressive query language – FOL with extensions. Relevant data is loaded into a resolution-based prover for inference. Pilot use in financial sector, as a part of a decision support system.

OntoGrate (U. of Oregon + Yale) Close to my approach: negative hyperresolution on Horn clauses is used to rewrite logical queries into SQL. Too ad hoc, missing a sound theoretical foundation (e.g., no completeness considerations).

Several projects leveraging the simplicity of inexpressive DL: E.g., traditional Chinese medicine DB integration project in Hangzhou (70 real DB!). Some form of query rewriting with SQL in the back end. Also, SHER (IBM) based on OWL-DL ABox summarisation (another form of abstraction). Case studies: trial cohort selection (RDB were pre-translated to OWL).

Summary

Hot real-life problem: deductive query answering over RDB.

The target is scalability: high expressiveness without performance sacrifice.

Original idea: incremental query rewriting with resolution and constraints.

Encouraging preliminary results: some formal basis, a prototype, preliminary case studies.