The Many Uses of Rules in Ontology-Based Data Access

(Long version: cs.unb.ca/~boley/talks/RulesOBDA.pdf)

Harold Boley
University of New Brunswick
Faculty of Computer Science
Fredericton, NB, Canada

Department of Informatics Colloquium, University of Zurich, Switzerland, April 10, 2014
11th Annual CS Research Exposition, UNB Fredericton, NB, May 2, 2014
Seminar, IBM Research, Zurich, Switzerland, June 23, 2014
FE Colloquium, WSL Birmensdorf, Switzerland, June 30, 2014
Kolloquium Informatik / Semantic Web Meetup, Freie Universität Berlin, Germany, July 4, 2014
Computational Logic Seminar - MUGS, Stanford Logic Group, Stanford University, CA, October 22, 2014
Seminar, Nuance Communications, Sunnyvale, CA, June 24, 2015
Seminar, Department of Computer Science, Stony Brook University, February 4, 2016
Abstract

Ontology-Based Data Access (OBDA) enables automated reasoning over an ontology as a generalized global schema for the data in local (e.g., relational or graph) databases reachable through mappings. OBDA can semantically validate, enrich, and integrate heterogeneous data sources. Motivated by rule-ontology synergies, this talk discusses key concepts of OBDA and their foundation in three kinds of (logical) 'if-then' rules, using examples from the DeltaForest case study on the susceptibility of forests to climate change. (1) A query is a special Datalog rule whose conjunctive body can be rewritten (see 2) and unfolded (see 3), and whose n-ary head predicate instantiates the distinguished answer variables of the body predicates. OBDA ontologies beyond RDF Schema (RDFS) expressivity usually permit negative constraints for data validation, which are translated to Boolean conjunctive queries corresponding to integrity rules. (2) The OBDA ontology supports query rewriting and database materialization through global-schema-level reasoning. It usually includes the expressivity of RDFS, whose class and property subsumptions can be seen as single-premise Datalog rules with, respectively, unary and binary predicates, and whose remaining axioms are also definable by rules. OBDA ontologies often extend RDFS to the description logic DL-Lite (as in OWL 2 QL), including subsumption axioms that correspond to (head-)existential rules. Recent work has also explored Rule-Based Data Access, e.g. via Description Logic Programs (as in OWL 2 RL, definable in RIF-Core), Datalog+/-, and Disjunctive Datalog. (3) OBDA data integration is centered on Global-As-View (GAV) mappings, which are safe Datalog rules allowing query unfolding of each global head predicate into a conjunction of local body predicates. These (heterogeneous) conjunctive queries can be further mapped to the database languages of the sources (e.g., to SQL or SPARQL). Conversely, the same GAV mappings allow database folding. The talk develops a unified architecture for (1) to (3).
What is Knowledge-Based Data Access?

- KBDA applies **AI / Semantic Technologies** to databases
- Founded on **(logical) knowledge** representation in, e.g., description logics or Datalog / deductive databases, query answering/optimization, and data integration/federation (e.g., Relational Logic Approach), with possible focus on:
  - **Ontology-Based Data Access (OBDA)**
  - **Rule-Based Data Access (RBDA)**

**Knowledge Base** (TBox, rule base) as generalized global schema for data in **local** — e.g., relational or graph — DBs (ABoxes, fact bases)

- KB module amplifies data storage & query execution of **distributed, heterogeneous** (No)SQL DBs
- Provides **multi-purpose** knowledge level for data

Explore synergies!
Preview: Unified KBDA Architecture (for Bidirectional Query Data Flow)

- Rewriting
- Materialization
- Folding Unfolding
- Mappings
- Q → DB’ → DB
- Global schema
- Local schemas
- Q’ → Q’ → DB’ → DB
- DB’ → DB
- DB → DB
- DB → DB
- DB → DB
Why Knowledge-Based Data Access?

- Domain knowledge utilized to deal with data torrent
- Domain experts conceptually *unfold* queries / *fold* data via **Mappings** defined with IT (SQL, SPARQL, ...) experts
- User concepts are captured in **Knowledge Base** for *domain-enriched* database querying / materialization without IT experts (queries also for data integrity)
- **Engines** use KB to deduce answers implicit in DBs
- **Analytics** enabled by queries exploring hypotheses
- **Enrichment** of KB with verified hypotheses
- KB as major organizational resource also for, e.g.:
  - Schema-level query answering (even without DBs)
KB Contains Formal Knowledge as Ontologies and/or Rules

FormalKnowledge

OntologyKnowledge

RuleKnowledge

TaxonomyKnowledge

FactKnowledge/Data
Data Access as Data Integration, Querying, and Management

Data Access conceived in layers:

- **Data Integration**: “read[+write]” – *combining* local (distributed, heterogeneous) data sources under a global schema
- **Data Querying**: “read-only” – *retrieving & inferring* answers from integrated data sources
- **Data Management**: “read+write” – *querying & updating* data sources (including with asserts and retracts)
Mediator vs. Warehouse Strategy in KBDA Architectures – Pros & Cons

**Mediator**: KB-enriched queries via *Query Rewriting*
- 😊 Less space in time/space trade-off (😊 more expansion, not execution time)
- 😊 Databases are left in original form (for big volume, variety, velocity)
- 😊 KB/DB separation makes all optimizations available from DB engines
- 😞 Repeated transformations for repeated queries (😊 workaround: caching)
- 😊 Partial views (😊 workaround: use maximally general, non-ground queries)

**Warehouse**: KB-enriched sources via *Database Materialization*
- 😊 Less time in DB time/space trade-off (😊 more space)
- 😊 Queries are left in original form
- 😞 Synchronization needed between DB original and copy
- 😞 Metadata often lost during DB transformation
- 😊 Total view (staging area for broad analytics)
Knowledge-Based Data Access, Using Strategy: 3-Dimensional KBDA$_s$

Knowledge

Rule

Ontology

Integration < Querying < Management

Strategy

bidirectional

mediator

warehouse
Three Dimensions of KBDA<sub>s</sub>: O,l,w

Ontology

- Based Data  Integration

Warehouse
Three Dimensions of $\text{KBDA}_s$: O,Q,w

Ontology

- Based Data

Querying

Warehouse
Three Dimensions of KBDA\textsubscript{s}: O,M,w

Ontology

- Based Data Management

Warehouse
Three Dimensions of KBDA$_s$: R, I, w

Rule-Based Data Integration
Three Dimensions of $\text{KBDA}_s$: $R, Q, w$

Rule RBDQ

Warehouse -Based Data Querying
Three Dimensions of $KBDA_s$: R, M, w
Three Dimensions of KBDA$_s$: O,l,m

Ontology

- Based Data

Integration
Three Dimensions of $\text{KBDA}_s$: O,Q,m

Ontology

OBDQ mediator

- Based Data Querying
Three Dimensions of $\text{KBDA}_s$: O,M,m

Ontology

OBDM

mediator

-Based Data Management
Three Dimensions of $\text{KBDA}_s$: $R, I, m$
Three Dimensions of $\text{KBDA}_s$: $R, Q, m$

Rule-based Data Querying

Rule $\text{RBDQ}$ mediator
Three Dimensions of $\text{KBDA}_s$: R, M, m

Rule-Based Data Management mediator
Three Dimensions of $\text{KBDA}_s$: bidirectional Knowledge-Based Data Access
Three Dimensions of $\text{KBDA}_s$: $R, Q, b$

Rule-Based Data Querying bidirectional
RBDA Realizes Uniform KBDA – 1 of 3: Queries as Rules

1. a) A conjunctive query is a special Datalog rule whose body can be rewritten (see 2.) and unfolded (see 3.), and whose head instantiates the distinguished answer variables of the body.

b) KBDA ontologies beyond RDF Schema (RDFS) often permit Boolean conjunctive queries corresponding to integrity rules.

2. ...

3. ...
RBDA Realizes Uniform KBDA – 2 of 3: KBs as Rules

1. ...

2. KBDA **KB** supports, e.g., query *rewriting* through global-schema-level reasoning, including with RDFS **taxonomies** or Datalog **rule** axioms, and DL-Lite (**OWL 2 QL**) or (**head-**)existential **rules**; KBDA **rules** also permit Description Logic Programs (**OWL 2 RL**), **Datalog**±, and **Disjunctive Datalog**. [Semantics of **ontology languages** customizable for expressivity and efficiency requirements by adding/deleting **rules** (**SPIN")]

3. ...
RBDA Realizes Uniform KBDA – 3 of 3: Mappings as Rules

1. ...
2. ...
3. KBDA data integration is centered on Global-As-View (GAV) mappings, which are Datalog rules for, e.g., unfolding each global head predicate to (a join, i.e. conjunction, of) local body predicates
RBDA Realizes Uniform KBDA — All of: Queries, KBs, and Mappings as Rules

1. a) A conjunctive query is a special Datalog rule whose body can be rewritten (see 2.) and unfolded (see 3.), and whose head instantiates the distinguished answer variables of the body
b) KBDA ontologies beyond RDF Schema (RDFS) often permit Boolean conjunctive queries corresponding to integrity rules

2. KBDA KB supports, e.g., query rewriting through global-schema-level reasoning, including with RDFS taxonomies or Datalog rule axioms, and DL-Lite (OWL 2 QL) or (head-)existential rules; KBDA rules also permit Description Logic Programs (OWL 2 RL), Datalog\textsuperscript{±}, and Disjunctive Datalog. [Semantics of ontology languages customizable for expressivity and efficiency requirements by adding/deleting rules (SPIN)]

3. KBDA data integration is centered on Global-As-View (GAV) mappings, which are Datalog rules for, e.g., unfolding each global head predicate to (a join, i.e. conjunction, of) local body predicates

This uniformity for mediators also holds for warehouses and bidirectionally...
Example: Forest/Orchard Knowledge
**EntityWithTree KB: Named Root Class (1)**

**Subsumption axioms (in description-logic syntax):**

EntityContainingAtLeastOneTree $\equiv$ Forest.

EntityContainingAtLeastOneTree $\equiv$ Orchard.

Forest $\equiv$ Woodland.

![Diagram of taxonomy tree with labels for EntityContainingAtLeastOneTree, Forest, Orchard, and Woodland.]

Root of taxonomy tree of tree-containing entities to see the forest for the trees

“$\equiv$” is taxonomy-style (description logic) ‘subsumes’ infix

Woodland
**EntityWithTree KB: Named Root Class (2)**

*Subsumption axioms (in higher-order rule syntax):*

EntityContainingAtLeastOneTree ← Forest.
EntityContainingAtLeastOneTree ← Orchard.
Forest ← Woodland.

Root of taxonomy tree of tree-containing entities to see the forest for the trees

"←" is rule-style ‘is implied by’ infix
EntityWithTree KB: Named Root Class (3)

Subsumption axioms (in higher-order rule syntax):
EntityContainingAtLeastOneTree :- Forest.
EntityContainingAtLeastOneTree :- Orchard.
Forest :- Woodland.

EntityContainingAtLeastOneTree

Forest

Orchard

Woodland

“←” is often used for logical implication rules.
“:- ” is often used for logic programming rules,
with premises allowing Negation as failure (Naf).
While we will not need Naf here, it can be mixed in
(e.g., as in the Relational Logic Approach and Naf Hornlog RuleML)

“:- ” is Prolog-style ‘if’ infix

Root of taxonomy tree of
tree-containing entities to see the forest for the trees
Subsumption axioms (in higher-order rule-infix syntax):

\[ \exists \text{contains.Tree} : - \text{Forest}. \]
\[ \exists \text{contains.Tree} : - \text{Orchard}. \]
Forest : - Woodland.

Cf. ontology-style (description logic) axioms:

\[ \exists \text{contains.Tree} \equiv \text{Forest} \]
\[ \exists \text{contains.Tree} \equiv \text{Orchard} \]

Entities each having a (multi-valued) contains property with at least one value in class Tree

C \equiv P is often written as P \equiv C.
C \leftarrow P is often written as P \rightarrow C.
C \vdash P is normally written only in this direction (in Prolog, Datalog, etc.).
Three Dimensions of $\text{KBDA}_s$: $R, Q, m$
Query Rewriting and Unfolding

- Mediator strategy uses:
  - KB to rewrite $Q$ to $Q'$ and Mappings (rules) to unfold $Q'$ to $Q_i$
  - KB can be ontology, e.g. in OWL 2 QL (DL-Lite), or rules
  - Abstract (relational/graph/...) queries $Q_i$ - grounded (to SQL/SPARQL/...) for $DB_i$
  - Each (relational/graph/...) database $DB_i$ left as original; answers at $\star$

Diagram:

- Query $Q$
- KB
- Rewriting
- Unfolding
- Mappings
- $Q'$
- $Q_1''$
- $Q_2''$
- $Q_n''$
- $DB_1$
- $DB_2$
- $DB_n$

Two 'backward' transformations
Query Rewriting and Unfolding

- Mediator strategy uses:
  - KB to *rewrite* $Q$ to $Q'$ and Mappings (rules) to *unfold* $Q'$ to $Q_i''$
  - KB can be *ontology*, e.g. in OWL 2 QL (DL-Lite), or **rules**
  - Abstract (relational/graph/...) queries $Q_i''$ ♦ -grounded (to SQL/SPARQL/...) for $DB_i$
  - Each (relational/graph/...) database $DB_i$ left as original; answers at ♦

![Diagram showing query rewriting and unfolding](image.png)
Query Rewriting Use of KB

- In Information Retrieval: Query expansion
  - With increased recall
  - Without loss of precision
- From Logic Programming (for Horn expressivity): Can use resolution method for KB-enrichment of a given (conjunctive) query with expanded (conjunctive) queries so that, for any DB, the answers to the enriched queries no longer using the KB are the same as the answers to the original query using the KB
Description Logic subsumptions (as higher-order rules):
∃contains.Tree :- Forest.
Forest :- Woodland.

Horn Logic rules (the first with conjunctive head):
(contains(?x s(?x)) ∧ Tree(s(?x))) :- Forest(?x).
Forest(?x) :- Woodland(?x).

Horn Logic rules (the first head split into two conjuncts):
contains(?x s(?x)) :- Forest(?x).
Tree(s(?x)) :- Forest(?x).
Forest(?x) :- Woodland(?x).
KB with Horn rules (from above) for rewriting of query rules:
contains(?x s(?x)) :- Forest(?x).
Tree(s(?x)) :- Forest(?x).
Forest(?x) :- Woodland(?x).

Rewriting Datalog query rule to obtain extra query rules:

Q: Given

q(?z) :- contains(?z ?y) ∧ Tree(?y).
q(?z) :- Forest(?z) ∧ Tree(s(?z)).
q(?z) :- contains(?z s(?x)) ∧ Forest(?x).
q(?z) :- Forest(?z) ∧ Forest(?z).
q(?z) :- Forest(?z).
q(?z) :- Woodland(?z).

Q’: Given ∪ Expansion
Query Unfolding Use of Mappings to Original Database Sources

- Datalog rules **bridging** between:
  - KB
  - Distributed DBs
- Use **partial deduction**-like unfolding (and simplification) of (conjunctive) KB queries to (conjunctive) *abstract DB queries*
  - Abstract relational queries grounded to SQL, abstract graph queries grounded to SPARQL, etc.
  - Lower-level optimization and execution by SQL, SPARQL, etc. engines
- Generated queries distributed over multiple DBs as indicated by “*source.*” name prefixes

Intuitive idea of **query unfolding**
Sample Mapping Rules to Three Local Data Sources

Map KB predicates to locDB/regionDB tables for geo data:
contains(?x ?y) :- locDB.cnt(?x ?kind ?y).
contains(?x ?y) :- regionDB.sub(?x ?r) ∧ locDB.cnt(?r ?kind ?y).

Map KB predicates to locDB/ecoDB tables for forestry data:
Tree(?t) :- locDB.cnt(?plot "tree" ?t).
Tree(?t) :- ecoDB.Plant(?plot "tree" ?size ?t).
Forest(?x) :- ecoDB.Habitat(?plot "forest" ?size ?x).
Woodland(?x) :- ecoDB.Habitat(?plot "wood" ?size ?x).
Mapping Rules Perform Unfolding of Rewritten Queries

**Union of conjunctive queries as Datalog rules (rewritten):**

\[ q(?z) :- \text{contains}(?z \ ?y) \land \text{Tree}(?y). \]
\[ q(?z) :- \text{Forest}(?z). \]
\[ q(?z) :- \text{Woodland}(?z). \]

**Unfolding above queries via mappings from previous slide:**

\[ q(?z) :- \text{locDB.cnt}(?z \ ?\text{kind} \ ?y) \land \text{locDB.cnt}(?\text{plot} \ "tree" \ ?y). \]
\[ q(?z) :- \text{locDB.cnt}(?z \ "tree" \ ?y). \]
\[ q(?z) :- \text{locDB.cnt}(?z \ ?\text{kind} \ ?y) \land \text{ecoDB.Plant}(?\text{plot} \ "tree" \ ?\text{size} \ ?y). \]
\[ q(?z) :- \text{regionDB.sub}(?z \ ?r) \land \text{locDB.cnt}(?r \ ?\text{kind} \ ?y) \land \text{locDB.cnt}(?\text{plot} \ "tree" \ ?y). \]
\[ q(?z) :- \text{regionDB.sub}(?z \ ?r) \land \text{locDB.cnt}(?r \ "tree" \ ?y). \]
\[ q(?z) :- \text{regionDB.sub}(?z \ ?r) \land \text{locDB.cnt}(?r ?\text{kind} \ ?y) \land \text{ecoDB.Plant}(?\text{plot} \ "tree" \ ?\text{size} \ ?y). \]
\[ q(?z) :- \text{ecoDB.Habitat}(?\text{plot} \ "forest" \ ?\text{size} \ ?z). \]
\[ q(?z) :- \text{ecoDB.Habitat}(?\text{plot} \ "wood" \ ?\text{size} \ ?z). \]
Local Data Source Consistency: Sample Integrity Rules (in 3 Forms)

Datalog rule with equality (in conjunctive head) for functionality constraints over source.table:

\( (?x_1 = ?x_2 \land ?k_1 = ?k_2) : \neg \text{locDB.cnt}(?x_1 ?k_1 ?y) \land \text{locDB.cnt}(?x_2 ?k_2 ?y). \)

Datalog rules with equality (head split into two conjuncts):

\( ?x_1 = ?x_2 : \neg \text{locDB.cnt}(?x_1 ?k_1 ?y) \land \text{locDB.cnt}(?x_2 ?k_2 ?y). \)
\( ?k_1 = ?k_2 : \neg \text{locDB.cnt}(?x_1 ?k_1 ?y) \land \text{locDB.cnt}(?x_2 ?k_2 ?y). \)

Same super-entity \( x_i \) and kind \( k_i \) if same sub-entity \( y \)

Permits Boolean queries

Datalog rules with inequality (in body) and falsity (in head):

\( \bot : \neg \text{locDB.cnt}(?x_1 ?k_1 ?y) \land \text{locDB.cnt}(?x_2 ?k_2 ?y) \land ?x_1 \neq ?x_2. \)
\( \bot : \neg \text{locDB.cnt}(?x_1 ?k_1 ?y) \land \text{locDB.cnt}(?x_2 ?k_2 ?y) \land ?k_1 \neq ?k_2. \)
Generalized Chaining-Mapping Rules

**Relational product of regionDB.sub and locDB.cnt (s. above):**
contains(?x ?y) :- locDB.cnt(?x ?kind ?y).
contains(?x ?y) :- regionDB.sub(?x ?r) \land locDB.cnt(?r ?kind ?y).

**Transitive closure of regionDB.sub based on locDB.cnt:**
contains(?x ?y) :- locDB.cnt(?x ?kind ?y).
contains(?x ?y) :- regionDB.sub(?x ?r) \land contains(?r ?y).

Recursive chaining
Three Dimensions of $\text{KBDA}_s$: R, Q, w

Rule-based Data Querying

Warehouse
Database Materialization after Folding

- Warehouse strategy uses:
  Mappings (same rules as for unfolding) to fold DB_i to DB and KB to *materialize* Database DB to DB’
- KB can be *ontology*, e.g. in OWL 2 RL (DLP), or *rules*
- Query is left as original; answers at solid triangular arrow head

(b)
Database Materialization after Folding

- Warehouse strategy uses:
  Mappings (same rules as for unfolding) to fold $DB_i$ to DB and KB to materialize Database DB to DB’
- KB can be ontology, e.g. in OWL 2 RL (DLP), or rules
- Query is left as original; answers at solid triangular arrow head
Database Folding Uses Mappings to Translate and Integrate Databases

- Datalog rules bridging between $n$ local $DB_i$ and 1 global $DB$
- Usually simpler than under mediator strategy
  - Materialization often done for higher-level $DB_i$, e.g. with Prolog facts as relational $DB_i$ or RDF triples as graph $DB_i$
  - Even if folding needs no table-to-facts or graph-to-triples translation, it still needs to do $n$-to-1 integration
- For $n=1$ and higher-level $DB_1$, no folding is needed at all (as assumed here)
Database Materialization Use of KB

- In Database Systems: Chase procedure
- From Logic Programming (for Horn expressivity): Can use bottom-up/forward-chaining method and KB for enrichment of a given set of facts (DB) with derived facts so that, for any query, the answers against the enriched DB’ no longer using the KB are the same as the answers against the original DB using the KB
EntityWithTree (EWT) KB

Subsumptions (as higher-order rules):

\[ \exists \text{contains.Tree} :\text{Forest} \]
\[ \text{Forest} :\text{Woodland} \]

\[ \exists \text{contains.Tree} \]

Forest

Woodland
EWT DB

Data (as higher-order facts):

\[ \exists \text{contains.Tree(e). Forest(f). Woodland(w)}. \]

DB: Original
EWT KB and DB: ‘Populated’ KB

Subsumptions and Data (as higher-order rules and facts):

∃contains.Tree :- Forest.
Forest :- Woodland.

∃contains.Tree(e).
Forest(f).
Woodland(w).

Diagram:

```
       ⊃contains.Tree
        ↓
       e
        ↓
    ⊃contains.Tree
     ↓
   f
  ↓
Woodland
  ↓
 w
```

Forest

Woodland
Applying All KB Rules to All DB Facts (1)

*Subsumptions and Data (as higher-order rules and facts):*

\[ \exists \text{contains.Tree} : \text{- Forest.} \]
\[ \text{Forest} : \text{- Woodland.} \]

\[ \exists \text{contains.Tree(e).} \]
\[ \exists \text{contains.Tree(f).} \]
\[ \text{Forest(f).} \]
\[ \text{Forest(w).} \]
\[ \text{Woodland(w).} \]
Applying All KB Rules to All DB Facts (2)

Subsumptions and Data (as higher-order rules and facts):

\[ \exists \text{contains.Tree} : \text{Forest}. \]
\[ \exists \text{contains.Tree} : \text{Woodland}. \]

\[ \exists \text{contains.Tree} : \text{Forest}(e). \]
\[ \exists \text{contains.Tree} : \text{Forest}(f). \]
\[ \exists \text{contains.Tree} : \text{Forest}(w). \]
\[ \exists \text{contains.Tree} : \text{Woodland}(w). \]
Fixpoint with No New Rule-Derived Facts

Subsumptions and Data (as higher-order rules and facts):

\[ \exists \text{contains.Tree} \ :- \ \text{Forest}. \]
\[ \text{Forest} :- \ \text{Woodland}. \]

\[ \exists \text{contains.Tree} \]
\[ \text{e} \quad \text{f} \quad \text{w} \]

Forest

Woodland

\[ \exists \text{contains.Tree(e)}. \]
\[ \exists \text{contains.Tree(f)}. \]
\[ \exists \text{contains.Tree(w)}. \]

Forest(f).
Forest(w).
Woodland(w).
Unpacking Conjunctions Implicit in Facts

**Subsumptions and Data (as higher-order rules and facts):**

\[
\exists \text{contains.Tree} : - \text{Forest.}
\]

\[
\text{Forest} : - \text{Woodland.}
\]

\[
\exists \text{contains.Tree(e).}
\]

\[
\text{contains(e s(e)).}
\]

\[
\text{Tree(s(e)).}
\]

\[
\exists \text{contains.Tree(f).}
\]

\[
\text{contains(f s(f)).}
\]

\[
\text{Tree(s(f)).}
\]

\[
\exists \text{contains.Tree(w).}
\]

\[
\text{contains(w s(w)).}
\]

\[
\text{Tree(s(w)).}
\]

\[
\text{Forest(f).}
\]

\[
\text{Forest(w).}
\]

\[
\text{Woodland(w).}
\]

\[
\exists \text{contains.Tree}.
\]

\[
\text{contains.}
\]

\[
\text{Tree}.
\]

\[
\text{Forest}.
\]

\[
\text{Woodland}.
\]

\[
\text{s is fresh Skolem function symbol}
\]

\[
\text{DB'}: \text{Materialized}
\]
Querying by Lookup in Materialized DB’

Datalog rules for (conjunctive) queries, with answers:

\[
\begin{align*}
q1(?z) & : \text{Woodland}(?z). & ?z = w \\
q2(?z) & : \text{Forest}(?z). & ?z = f, w \\
q3(?z) & : \exists \text{contains.Tree}(?z). & ?z = e, f, w \\
q4(?z) & : \text{contains}(?z \ ?y) \land \text{Tree}(?y). & ?z = e, f, w
\end{align*}
\]

Separate conjunctive queries qi: After materialization, no need for query rewriting with the same KB.
Three Dimensions of $\text{KBDA}_s$: $R, Q, b$
Query Rewriting Along with Database Materialization after Folding (General)

- Bidirectional strategy (Mediator-Warehouse combination)
- Both directions traversable concurrently (DB’ as sync point)
- Answers at solid triangular arrow head
Query Rewriting Along with Database Materialization after Folding (General)

- Bidirectional strategy (Mediator-Warehouse combination)
- Both directions traversable concurrently (DB’ as sync point)
- Answers at solid triangular arrow head
Query Rewriting Along with Database Materialization after Folding (Simplified)

- Bidirectional strategy for n=1:
  \( \text{KB}^{\text{Upper}} \) disjoint from \( \text{KB}^{\text{Lower}} \) (bipartitioning of KB)

(c₁)

\[
\begin{align*}
\text{Q} & \xrightarrow{\text{Rewriting}} \text{DB'} \xrightarrow{\text{Materialization}} \text{DB} \xrightarrow{\text{Folding}} \text{DB}_1 \\
\text{Q'} & \xrightarrow{\text{Folding}} \text{DB}\xrightarrow{\text{Materialization}} \text{DB'} \xrightarrow{\text{Rewriting}} \text{Q}
\end{align*}
\]
Query Rewriting Along with Database Materialization after Folding (Simplified)

- Bidirectional strategy for $n=1$: $\text{KB}^{\text{Upper}}$ disjoint from $\text{KB}^{\text{Lower}}$ (bipartitioning of KB)
Partial Query Rewriting
Using Part of Bipartitioned KB

- Can use resolution method on part of KB for enrichment of a given (conjunctive) query with expanded (conjunctive) queries so that, for any DB, the answers to the enriched queries only using the other (complement) part of the KB are the same as the answers to the original query using the original KB

- In the following, KB = KB^{Upper} ⊕ KB^{Lower} will be bipartitioned into Upper and Lower part, here using KB^{Upper}
Rule Clausification of Upper EWT KB

*Upper Description Logic subsumption (as higher-order rule):*

\[ \exists \text{contains.Tree} : \text{Forest}. \]

Higher-order syntactic sugar for existential rule:

\[ \exists \text{s.contains}(\text{x } \text{?y}) \land \text{Tree(} \text{?y} \text{)} : \text{- Forest(} \text{?x} \text{).} \]

*Horn Logic rule (with conjunctive head):*

\[ \text{contains(} \text{?x } \text{s(} \text{?x} \text{)) } \land \text{Tree(} \text{s(} \text{?x} \text{))} : \text{- Forest(} \text{?x} \text{).} \]

`s` is fresh Skolem function symbol

*Horn Logic rules (head split into two conjuncts):*

contains(\text{x } \text{s(} \text{?x} \text{)}): \text{- Forest(} \text{?x} \text{).}

Tree(\text{s(} \text{?x} \text{)}): \text{- Forest(} \text{?x} \text{).}
Upper KB Rules Perform Rewriting of Query

**KB with Horn rules (from above) for rewriting of query rules:**
- `contains(?x s(?x)) :- Forest(?x).`
- `Tree(s(?x)) :- Forest(?x).`

---

**Rewriting Datalog query rule to obtain extra query rule:**
- `q(?z) :- contains(?z ?y) ^ Tree(?y).`
- `q(?z):-Forest(?z)^Tree(s(?z))`  
- `q(?z):-contains(?z s(?x))^Forest(?x)`  
- `q(?z) :- Forest(?z) ^ Forest(?z).`
- `q(?z) :- Forest(?z).`

---

*Q: Given*  
*Expansion*

*Q*: Given $\cup$ Expansion
Partial Database Materialization Using Part of Bipartitioned KB

- Can use **bottom-up/forward-chaining method** on part of KB for enrichment of a given set of facts (DB) with derived facts so that, for any query, the answers against the enriched DB’ only using the other (complement) part of the KB are the same as the answers against the original DB using the original KB.

- In the following, KB will again be bipartitioned into Upper and Lower part, here using $KB^{\text{Lower}}$. 
Lower EWT KB and Unchanged DB

Lower Subsumption and original Data \( (as \ higher-order\ rule\ and\ facts):\)

Forest :- Woodland.

\( \exists \) contains.Tree(e).
Forest(f).
Woodland(w).

\( DB = DB_1: \) Original
Applying the KB Rule to All DB Facts

Lower Subsumption and original Data (as higher-order rule and facts):

Forest :- Woodland.

\exists \text{contains}. \text{Tree}(e).

Forest(f).

Forest(w).

Woodland(w).
Fixpoint with No New Rule-Derived Facts

Lower Subsumption and original Data (as higher-order rule and facts):

\[
\text{Forest} \leftarrow \text{Woodland}.
\]

\[
\exists \text{contains.Tree(e)}.
\]

\[
\text{Forest(f)}.
\]

\[
\text{Forest(w)}.
\]

\[
\text{Woodland(w)}.
\]
Unpacking Conjunction Implicit in Fact

Lower Subsumption and original Data (as higher-order rule and facts):

\[ \text{Forest} :- \text{Woodland}. \]

\[ \exists \text{contains.} \text{Tree}(e). \]

\[ \text{contains}(e \ s(e)). \]

\[ \text{Tree}(s(e)). \]

\[ \text{Forest}(f). \]

\[ \text{Forest}(w). \]

\[ \text{Woodland}(w). \]

\[ s \text{ is fresh Skolem function symbol} \]

\[ \text{DB'}: \text{Materialized} \]
Querying by Lookup in Materialized DB’
(cf. Slide “Upper KB Rules Perform Rewriting of Query”)

Datalog rules for (conjunctive) queries, with answers:

\[ q(?z) : \text{contains}(?z \ ?y) \land \text{Tree}(?y). \quad ?z = e \]
\[ q(?z) : \text{Forest}(?z). \quad ?z = f, w \]

Union of conjunctive queries q:
- partial query rewriting
- and partial materialization
- using parts (Upper, Lower) of the same KB
Three Dimensions of $\text{KBDA}_s$

- Knowledge
- KBDA
- Strategy
- Based Data
- Access
Query Rewriting and Unfolding: Recap

(a) $Q \xrightarrow{\text{Rewriting}} \text{KB} \xrightarrow{\text{Mappings}} \{Q_1, Q_2, \ldots, Q_n\} \xrightarrow{\text{Unfolding}} \{DB_1, DB_2, \ldots, DB_n\}$
Database Materialization after Folding: Recap

(b)

Q

DB'

Materialization

Folding

Mappings

DB

DB_1

DB_2

\ldots

DB_n
Query Rewriting Along with Database Materialization after Folding: Interlude
Unified Architecture

- Combines strategies (a)-(c) of earlier slides
- Meets the needs of \( \Delta \)Forest case study
Unified Architecture

- Combines strategies (a)-(c) of earlier slides
- Meets the needs of ΔForest case study
Standard Language for KBDA$_s$ Rules

- Will permit (Web-based) interchange of at least the three kinds of rules in KBDA$_s$:
  - Sharing and Reuse of queries (+ integrity constraints), KBs, and mappings can save repeated work

- Language options include
  - ISO: Prolog (extra-logical features), Common Logic 2 (‘wild-west’ syntax)
  - W3C: OWL 2 RL (DLP: weak), RIF-BLD (Horn: no head-$\exists$)
  - OMG: PRR (metamodel only), SBVR (mostly Controlled English)
  - RuleML: Deliberation RuleML (customizable expressivity)
  - Flora-2, Ergo: Rulelog (extends database expressivity)
RuleML System of Families of Languages

- RuleML system covers a wide rule spectrum: Deliberation Family (1.02) ... Reaction Family (1.02)
  - Rule condition part shared across the spectrum
  - Syntactic uniformity allows wide reuse (XML, JSON)
- System now constitutes a deep language lattice
  - Major language inclusion path:
    Deliberation $\supset$ HOL $\supset$ FOL $\supset$ Derivation $\supset$ Hornlog $\supset$ Datalog $\supset$ ...
- Naf mix-in customization of Hornlog RuleML (Naf Hornlog RuleML) leads to Logic Programs
**RBDA\textsubscript{s}-Style KBDA\textsubscript{s} Architecture: Expressivity of Rule Systems**

- The language of the global schema can be generalized from unary/binary (OBDA\textsubscript{s}) to n-ary predicates (RBDA\textsubscript{s})
- When decidability of querying is not required, RBDA\textsubscript{s} expressivity can be extended from Datalog, Datalog\textsuperscript{\pm}, and description logics to Datalog\textsuperscript{+}, Horn logic, and FOL, as enabled by Deliberation RuleML 1.02,
- Features customizable with the MYNG 1.02 GUI
- Moreover, Reaction RuleML 1.02 can express updates, as needed for KBDM\textsubscript{s} (Ontology-based Data Management)
**RBDA$_S$-Style KBDA$_S$ Architecture: Uniformity via Rule Systems**

- Rule-based style of Unified Architecture (earlier slide)
- Presentation syntax (":-"), serialization ("<RuleML>"), and semantics approach (model theory) uniform from queries (+ integrity constraints) to KBs to mappings to abstract DBs
- Division of labor between KB rules and mapping rules can be modified *without crossing paradigm boundaries*
  - Allows KB- and mapping-directed normal forms
- Assumptions (unique-name and closed-world) of DBs and many rule systems made explicit by *semantic styles*
Forest: Study Overview

- RuleML provides a family of rule (incl. fact/data & query) languages of customizable expressivity, a family-uniform XML format, and a suite of (MYNG) tools for processing

- WSL creates knowledge and publishes data about Swiss forests, giving an integrated federal perspective on heterogeneous databases of various (geographically, thematically, ...) specialized sources

- RuleML-WSL collaboration has brought RuleML technology to bear on WSL knowledge and DBs for RBDA

- Prepared data sets, defined global schema and mappings, formalized knowledge, and specialized RBDA architecture in support of complex statistical data analysis
Forest: Study Team

- Int’l collaboration between: RuleML, WSL, UNB
- Participants:
  - Harold Boley (UNB, RuleML, WSL[visiting Jan-Jun 2014])
  - Rolf Grütter (WSL)
  - Tara Athan (RuleML)
  - Gen Zou (UNB)
  - Sophia Etzold (WSL)
Forest: Three Local Data Sources

- Productivity Research Areas: Ertragskundeflächen (EKF)
  - 83 areas
  - Pure stands
  - Approximate 10-year intervals (time series of different lengths)
  - Data: .dbf/.csv files (ca. 100 MB)
  - 3 tables

- Natural Forest Reserves: Naturwaldreservate (NWR)
  - 36 areas
  - Pure and mixed stands
  - 10-year intervals
  - Data: .txt files (ca. 65 MB)
  - 1 file per area

- Long-term Forest Ecosystem Research: Langfristige Waldökosystemforschung (LWF)
  - 18 areas
  - Pure and mixed stands
  - 5-year intervals (since 1995)
  - Data: Oracle DB (ca. 500 GB)
  - RDB tables
**Global schema**

**Forest:** Schemas for Architecture (a)-(d)

```
Global schema
```

```
△Forest: Schemas for Architecture (a)-(d)
```

```
DB'
```

```
PlotsStatic
- plot
- source
- x
- y
- altitude
- class
```

```
PlotsDynamic
- plot
- year
- age
- dmg
- dg
- n
```

```
SGAbundance
- plot
- species-group
- percentage
```

```
RelMortality
- plot
- year1
- year2
- relmortality
```

**Local schemas**

```
Local schemas
```

```
EKF
```

```
NWR
```

```
LWF
```

```
Keys – three composite – in **bold red**
```
Eligibility Criteria

1. Impact of **forest management** on tree mortality is negligible at the investigated sites
   - NWR: impact is negligible by definition
   - EKF: impact of grades A, B is negligible (≠ C, D, H, P)
   - LWF: not recorded

2. Plots are **statistically independent** of each other
   - Plots that are located within a distance of 500 meters from each other are possibly dependent

3. Plots contain a **time-averaged abundance** greater than a threshold value for at least one of the target species
Questions Addressed

1. Are there sufficiently many eligible plots in order to perform an analysis per main tree species?
2. Are there sufficiently many eligible plots in order to perform an analysis per main tree species and climatic region?
3. Which eligible plots represent pure tree stands and which eligible plots represent mixed tree stands?
Query Rewriting

\[
q(?plot) :- \text{EligiblePlot}(?plot) \\
\text{TreeStandKey}(?id \ ?plot \ "oak") \\
\text{TreeStandAbundance}(?id \ ?pct) \\
?pct >= 15.
\]

\[
\exists \text{id} (\text{TreeStandKey}(\text{id} \ ?plot \ ?sp) \ \text{TreeStandAbundance}(\text{id} \ ?pct) :- \\
\text{TreeStandMerged}(?plot \ ?sp \ ?pct).
\]

\[
q(?plot) :- \text{EligiblePlot}(?plot) \\
\text{TreeStandMerged}(?plot \ "oak" \ ?pct) \\
?pct >= 15.
\]
Query Unfolding

\[
\text{q2(?plot)} :\text{- TreeStandMerged(?plot "oak" ?pct)} \\
\quad ?pct \geq 15.
\]

\[
\text{TreeStandMerged(?plot "oak" ?pct)} :\text{-} \\
\quad \text{EKF.dom(?plot "Quercus petraea" ?pct1)} \\
\quad \text{EKF.dom(?plot "Quercus robur" ?pct2)} \\
\quad ?pct = ?pct1 + ?pct2.
\]

\[
\text{q2(?plot)} : \text{- EKF.dom(?plot "Quercus petraea" ?pct1)} \\
\quad \text{EKF.dom(?plot "Quercus robur" ?pct2)} \\
\quad ?pct1 + ?pct2 \geq 15.
\]
Conclusions

- Ontology-Based Data Access (OBDA) founded on three kinds of rules: *Query rules* (including integrity rules), *KB rules* (for query rewriting and DB materialization), as well as *mapping rules* (for query unfolding and DB folding).
- OBDA complemented by Rule-Based Data Access (RBDA) and generalized to Knowledge-Based Data Access (KBDA).
- Specified an RBDA-uniform KBDA\(_s\) architecture with unified mediator, warehouse, and bidirectional strategies.
- RuleML used for XMLserialized rules, MYNG-customized rule expressivity, and platform-independent RBDA.
- Introduced \(\Delta\text{Forest}\) specialization of RBDA architecture for statistical data analysis in ecosystem research at WSL.
Future Work (1)

- Translate simplified presentation syntax into pre-released XML serialization of Deliberation RuleML 1.02 / MYNG 1.02
- Support implementations of specified architecture reusing (open source) KBDA technology (cf. RBDA wiki page)
- For high-precision RBDQS language support, complement current techniques of Datalog\(^{+}\) RuleML with Datalog\(^{-}\) RuleML using context-sensitive/semantic validators for “-” constraints
- Evaluate (mediator/warehouse, relational/graph, ...) trade-offs for KBDQS in PSOA RuleML as executed in PSOATransRun
- Develop ΔForest study at WSL for extended and new data sources of big volume, variety, and velocity (e.g., about climate change)
- Augment geospatial KBDQS mappings with Optique mapping (bootstrapping, repair, ...) techniques
Future Work (2)

- Compare engines for \( \text{OBDQ}_s \) and \( \text{RBDQ}_s \), including HYDRA and RDFox, w.r.t. expressivity and efficiency
- Adapt Semantic Automated Discovery and Integration (SADI) test cases for \( \text{KBDQ}_s \) experiments in PSOA RuleML querying
- Evaluate Abstract Logic-based Architecture Storage systems & Knowledge base Analysis (ALASKA) for \( \text{RBDA}_s \)
- Extend the \( \text{KBDA}_s \) architecture with semantic annotation rules for (Deep) Web data extraction (Deep Web Mediator)
- Use Grailog for \( \text{KBDA}_s \) data and knowledge visualization
- Explore synergies between the logical \( \text{KBDA}_s \) approach with statistical approaches, e.g. from Statistical Relational AI
BACKUP Slides
What Kinds of Jeopardy! Queries (from IBM Watson’s 2011 First Prize Game) are Addressed Here?

- Queries that are non-metaphorical
  - “To push one of these paper products is to stretch established limits” (“the envelope”, Brad Rutter)

- Queries not asking for a logical predicate/relation
  - “To bring back someone to his original function or position” (“reinstate”, Ken Jennings)

- Queries asking for a logical individual constant
  - “It's New Zealand's second-largest city” (“Christchurch”, IBM Watson)
What Logic Do the Addressed *Jeopardy!* Queries Adhere to?

- Function-free Horn logic / Pure Prolog (Datalog)
- *Jeopardy!*
  “It's New Zealand's second-largest city”
- Paraphrase (with a suggestive Wikipedia link)
  “Which city is New Zealand's second largest urban area w.r.t. population ranking?”
- Formalization (as a query rule with variable \(?z\))
  \[
  q(?z) :- \text{city}(?z) \land \text{urban_pop_rank}(\text{New Zealand} 2 ?z).
  \]
KBDA\textsubscript{s} \textit{(e.g. OBDQ\textsubscript{mediator})} Mappings

- Can be
  - relational-to-relational
  - relational-to-graph \textit{(e.g., SQL to SPARQL)}
  - graph-to-relational
  - graph-to-graph
Rule Languages

- Paradigms for modeling entity dependencies:
  - Relational
  - Graph (Object-Centered)
  - Combined

- Since Knowledge Bases (KBs) have been developed in languages following all three paradigms, cross-paradigm translation, integration, and reuse is often necessary.

- Need for an interoperation language and technology: Positional-Slotted Object-Applicative (PSOA) RuleML

- Naturally combinable with portability technology: Platform-independent implementation of PSOA RuleML
Hypergraph Example – Relational Betweenness

Directed hyperarcs cut through intermediate nodes (cf. Grailog)

Facts

betweenRel(pacific, canada, atlantic)
betweenRel(canada, usa, mexico)
"#" denotes "∈" for class membership; "→" associates a slot name with its filler

\[ b0 \#\text{betweenObj}(\text{outer1} → \text{pacific}; \text{inner} → \text{canada}; \text{outer2} → \text{atlantic}) \]
\[ b1 \#\text{betweenObj}(\text{outer1} → \text{canada}; \text{inner} → \text{usa}; \text{outer2} → \text{mexico}) \]
Example – Integrated Betweenness (Enriched)

betweenObjRel

westEast

orient

b0

dim

2

dim

b1

orient

pacific

canada

atlantic

usa

mexico

northSouth

Facts

\( b0 \# \text{betweenObjRel}(\text{pacific, canada, atlantic}; \ dim \rightarrow 2; \ orient \rightarrow \text{westEast}) \)

\( b1 \# \text{betweenObjRel}(\text{canada, usa, mexico}; \ dim \rightarrow 2; \ orient \rightarrow \text{northsouth}) \)
Relational Rule Languages

- Widely used for relational DBs (SQL views) and KBs, representing information in classical logic
- Model dependencies among $n$ entities as an $n$-ary predicate applied to an ordered sequence of $n$ arguments, called positional arguments
- Languages: Common Logic, Prolog, TPTP-FOF, ...
Graph (Object-Centered) Rule Languages

- Receive increasing attention because of expanding research and development in linked data on the Web, graph/‘triple’ stores, and big data in NoSQL DBs.
- Each object is represented by a unique Object IDentifier (OID), typed by a class, and described by an unordered collection of slots, each being a pair of a name and a filler.
- An OID-describing slotted term in AI is called a frame (represents a resource/‘subject’-describing property list on the Semantic Web).
- Languages: RDF, N3, ...
Object-Relational Rule Languages

- Combine the object-centered and relational paradigms, either in a heterogeneous or a homogeneous way
- Heterogeneous
  - Allow atomic formulas in both relational and object-centered forms, even mixed in the same rule
  - Languages: F-logic and RIF
- Homogeneous
  - Integrate relational and object-centered atomic formulas into a unified form
  - Language: PSOA RuleML
PSOA RuleML

- Integrates relational and object-centered modeling
- Generalizes F-logic, RIF-BLD, and POSL
- Uses positional-slotted object-applicative (psoa) terms, permitting a relation application to have an OID – typed by the relation – and, orthogonally, its arguments to be positional or slotted.

General case (multi-tuple):
\[
o\#f([t_{1,1} \ldots t_{1,n_1}] \ldots [t_{m,1} \ldots t_{m,n_m}] \ p_1\rightarrow v_1 \ldots p_k\rightarrow v_k)
\]

Special cases (single-tuple brackets and zero-argument parentheses optional):
- Combined: \[
o\#f([t_1 \ldots t_n] \ p_1\rightarrow v_1 \ldots p_k\rightarrow v_k)
\]
- Positional: \[
o\#f([t_1 \ldots t_n])
\]
- Slotted: \[
o\#f( \ p_1\rightarrow v_1 \ldots p_k\rightarrow v_k)
\]
- Member-only: \[
o\#f()
\]
Example of Querying a PSOA Fact and Rule

KB:

\[ b1\#betweenObjRel(\text{canada usa mexico}) \]
\[ \quad \text{dim}\rightarrow2 \quad \text{orient}\rightarrow\text{northSouth} \]

Forall ?out1 ?in ?out2 ?b

\[
\quad ?\text{in}\#GeoUnit(\text{neighborNorth}\rightarrow?\text{out1})
\]
\[
\quad \text{neighborSouth}\rightarrow?\text{out2}) \quad :-
\]
\[
\quad ?b\#betweenObjRel(?\text{out1} \quad ?\text{in} \quad ?\text{out2})
\quad \text{orient}\rightarrow\text{northSouth})
\]

English Query: “Which GeoUnit has Canada as its northern neighbor?”
Query: \(?X\#GeoUnit(\text{neighborNorth}\rightarrow\text{canada})\)
Answer: \(?X=\text{usa}\)

TPTP version in Common Logic would contribute to COLORE