VADALOG
A Swift Logic for Big Data
and a System Combining Datalog Reasoning with Machine Learning

Georg Gottlob
Univ. of Oxford and TU Wien
Reasoning tasks are required that cannot be expressed by description logics, and cannot be reasonably managed by relational DBMS, nor by graph DBMS.
Example: Wikidata Marriage Intervals

[Krötzsch DL 2017]

Wikidata contains the statement:

Taylor was married to Burton starting from 1964 and ending 1974

This can be represented in relational DB or Datalog-notation by:

married(taylor, burton, 1964, 1974)

Symmetry rule for marriage intervals in Datalog:

∀ u, v, x, y. married(u, v, x, y) → married(v, u, x, y)

This cannot be expressed in DLs!

Note: In what follows, we will often omit universal quantifiers.
Example: Controlling Companies
Example: Controlling Companies

x controls y if
x directly holds over 50% of y, or
x controls a set of companies that jointly hold over 50% of y

This cannot be expressed in DLs and only clumsily in SQL and Graph DBMS!
Example: My Creditworthiness
Example: My Creditworthiness
My Explanation

A machine-learning program has “reasonably” learned:

People who live in a joint household with someone who does not pay their bills are likely to fail repaying their own debts.

This ethically questionable rule was applied to incomplete and wrong data:

• Before I bought the house there was a tenant who indeed did not pay his bills (tons of unpaid bills & overdue notices in my mailbox).

• The tenant had moved out before I moved in, but the Credit Rating Agency did not know, and simply assumed he still lived there.
ML should be complemented and, where necessary, overruled by domain-specific “expert rules” that express domain knowledge:

- A new house-owner is most likely unrelated to a previous tenant.

- If a house is bought by somebody who did not live there previously, and now lives there, then the previous occupiers have most likely moved out. 
  [→Verify!]

- If someone has closed their bank account without opening a new one then it is likely that the person has moved out of the country.

... 

Automatically accessing outside sources such as the Land Register and/or Social Networks may help.

→ Data extraction from external sources is a requirement for KGMS.
Knowledge Graph Management Systems

KGMS combine the power of rule-based reasoning with machine learning over Big Data:

\[ \text{KGMS} = \text{KBMS} + \text{Big Data} + \text{Analytics} \]

Misusing the lateralization thesis for illustration
Knowledge Layers

Vertical-specific Knowledge Layers
- Logistics
- Banking & Finance
- Oil & Gas
- Media Intel
- Life Sciences
- ...
- ...

General Knowledge Layers
- Graph Analytics
- Data Wrangling
- Web Data Extraction
- Entity Resolution

Core Reasoning Engine
- Strong performance and Expressiveness, Graph Navigation,
- Integrations with Machine Learning & Enterprise Databases
Vadalog: The Core Reasoning Language

Core Vadalog = full Datalog + restricted use of $\exists$ + stratif. negation + $\bot$

Why existential quantifiers in rule heads?

- Data exchange, data integration
- Data extraction
- Reasoning with RDF $\rightarrow$ Wikidata example
- Ontology querying (DL-Lite, EL, etc.)
- Automated product configuration
- Conceptual Modeling (e.g., UML)
Data Exchange, Data Provisioning, Data Wrangling

Source Schema $S$

Target Schema $T$

$\exists W \ \text{person}(Y,X,W)$

[Fagin, Kolaitis, Miller & Popa, 2003]; [Arenas et al., 2014]
<table>
<thead>
<tr>
<th>PRODUCT</th>
<th>PRICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Toshiba_Protege_cx</td>
<td>480</td>
</tr>
<tr>
<td>Dell_25416</td>
<td>360</td>
</tr>
<tr>
<td>Dell_23233</td>
<td>470</td>
</tr>
<tr>
<td>Acer_78987</td>
<td>390</td>
</tr>
<tr>
<td>PRODUCT</td>
<td>PRICE</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------</td>
</tr>
<tr>
<td>Toshiba_Protege_cx</td>
<td>480</td>
</tr>
<tr>
<td>Dell_25416</td>
<td>360</td>
</tr>
<tr>
<td>Dell_23233</td>
<td>470</td>
</tr>
<tr>
<td>Acer_78987</td>
<td>390</td>
</tr>
</tbody>
</table>
we need object creation...

<table>
<thead>
<tr>
<th>PRODUCT</th>
<th>PRICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Toshiba_Protege_cx</td>
<td>480</td>
</tr>
<tr>
<td>Dell_25416</td>
<td>360</td>
</tr>
<tr>
<td>Dell_23233</td>
<td>470</td>
</tr>
<tr>
<td>Acer_78987</td>
<td>390</td>
</tr>
</tbody>
</table>
Data Extraction

table(T₁),
table(T₂),
sameColor(T₁,T₂),
isNeighbourRight(T₁,T₂) →

\[ \exists T \text{ tablebox}(T), \]
\[ \text{contains}(T,T₁), \]
\[ \text{contains}(T,T₂). \]

<table>
<thead>
<tr>
<th>PRODUCT</th>
<th>PRICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Toshiba_Protege_cx</td>
<td>480</td>
</tr>
<tr>
<td>Dell_25416</td>
<td>360</td>
</tr>
<tr>
<td>Dell_23233</td>
<td>470</td>
</tr>
<tr>
<td>Acer_78987</td>
<td>390</td>
</tr>
</tbody>
</table>
Reasoning with **RDF** – Object Creation

married(taylor, burton, 1964, 1974)

In the **RDF**-like “graph” notation this tuple is broken up into several triples (here represented as logical facts):

\[
\begin{align*}
\text{spouse1}(k1, taylor), \\
\text{spouse2}(k1, burton), \\
\text{start}(k1, 1964), \\
\text{end}(k1, 1974)
\end{align*}
\]

This symmetry rule for marriage intervals now becomes:

\[
\forall u, v, x, y. \text{married}(u, v, x, y) \rightarrow \text{married}(v, u, x, y)
\]

∀ u, v, x, y. married(u, v, x, y) → married(v, u, x, y)

This symmetry rule for marriage intervals now becomes:

\[
\begin{align*}
\text{spouse1}(u, y1) \land \text{spouse2}(u, y2) \land \text{start}(u, y3) \land \text{end}(u, y4) \rightarrow \\
\exists v. \text{spouse}(v, y1) \land \text{spouse1}(v, y2) \land \text{start}(v, y3) \land \text{end}(v, y4)
\end{align*}
\]

\[
\begin{align*}
\text{spouse1}(k2, burton), \\
\text{spouse2}(k2, taylor), \\
\text{start}(k2, 1964), \\
\text{end}(k2, 1974)
\end{align*}
\]
Description Logics & Ontological Reasoning

The DL-Lite Family

Popular family of DLs with low ($AC_0$) data complexity

<table>
<thead>
<tr>
<th>DL-Lite TBox</th>
<th>First-Order Representation (Datalog$^\pm$)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DL-Lite$_{core}$</strong></td>
<td></td>
</tr>
<tr>
<td>$professor \sqsubseteq \exists \text{teachesTo}$</td>
<td>$\forall X , professor(X) \rightarrow \exists Y , \text{teachesTo}(X,Y)$</td>
</tr>
<tr>
<td>$professor \sqsubseteq \neg \text{student}$</td>
<td>$\forall X , professor(X) \land \text{student}(X) \rightarrow \bot$</td>
</tr>
<tr>
<td><strong>DL-Lite$_R$ (OWL 2 QL)</strong></td>
<td></td>
</tr>
<tr>
<td>$\neg \text{hasTutor} \sqsubseteq \text{teachesTo}$</td>
<td></td>
</tr>
<tr>
<td><strong>DL-Lite$_F$</strong></td>
<td></td>
</tr>
<tr>
<td>$\text{funct}(\text{hasTutor})$</td>
<td>$\forall X \forall Y \forall Z , \text{hasTutor}(X,Y) \land \text{hasTutor}(X,Z) \rightarrow Y = Z$</td>
</tr>
</tbody>
</table>

Datalog[$\exists$]: Full Datalog augmented with $\exists$-quantifier

Unfortunately:

**Theorem:** Reasoning ($KB \models q$) with Datalog[$\exists$] is undecidable.

[Beeri & Vardi, 1981]; [J. Mitchell 1983] [Chandra & Vardi 1985];
[Cali, G., & Kifer, 2008]; [Baget, Leclère & Mugnier, 2010]

Finding expressive decidable fragments has become a topic of intensive research over the last 10 years.

**Datalog$^{\pm}$:** Datalog[$\exists, \bot, \neg$-strat, ...] subject to syntactic restrictions.

**Vadalog:** member of the Datalog$^{\pm}$ family admitting efficient reasoning methods.
Recall Marie-Laure Mugnier’s talk
Main Decidable Datalog$^\pm$ Languages

- Guarded
- Sticky
- Weakly-(frontier-) guarded
- Linear
- Core VADALOG
- DATALOG
- DL-Lite$_R$
- $\mathcal{ELHI}$
- BTS
- Weakly-Sticky
- FUS
Datalog\[∃,\ldots\]

UNDECIDABLE

weakly frontier-guarded Datalog\[∃,⊥,¬\text{strat}\]

EXPTIME

Core Vadalog

= warded Datalog\[∃,⊥,¬\text{strat}\]

PTIME

Datalog\[⊥,¬\text{strat}\]

Linear Datalog\[^{±}\[∃,⊥\]]\]

SPARQL + OWL 2QL

Data complexity

\(\Delta^0\)
Datalog[$\exists,...]$  

UNDECIDABLE

weakly frontier-guarded Datalog[$\exists,\perp,\neg \text{strat}$]  

EXPTIME

Core Vadalog  

= warded Datalog[$\exists,\perp,\neg \text{strat}$]

PTIME  

NLOGSPACE

Data complexity
Affected Positions in a Datalog\(^\pm\) Program

Those positions of a predicate, where existential values (Skolem terms) can “flow in”

\[
P(X,Y), S(Y,Z) \rightarrow \exists W \ T(Y,X,W) \]
\[
T(X,Y,Z) \rightarrow \exists W \ P(W,Z) \]
\[
P(X,Y) \rightarrow \exists Z \ Q(X,Z) \]
Affected Positions in a Datalog\(^{\pm}\) Program

Those positions of a predicate, where existential values (Skolem terms) can “flow in”

\[ P(X,Y), S(Y,Z) \rightarrow \exists W \ T(Y,X,W) \]
\[ T(X,Y,Z) \rightarrow \exists W \ P(W,Z) \]
\[ P(X,Y) \rightarrow \exists Z \ Q(X,Z) \]
Affected Positions in a Datalog$^\pm$ Program

Those positions of a predicate, where existential values (Skolem terms) can “flow in”

\[
\begin{align*}
P(X,Y), S(Y,Z) &\rightarrow \exists W T(Y,X,W) \\
T(X,Y,Z) &\rightarrow \exists W P(W,Z) \\
P(X,Y) &\rightarrow \exists Z Q(X,Z)
\end{align*}
\]

Affected Positions

T[3], P[1], Q[2]

T[2], P[2], Q[1]
Dangerous Variables in Rule Bodies

Head variables that, in the body, occur only in affected positions.

\[ P(X, Y), S(Y, Z) \rightarrow \exists W \; T(Y, X, W) \]
\[ T(X, Y, Z) \rightarrow \exists W \; P(W, Z) \]
\[ P(X, Y) \rightarrow \exists Z \; Q(X, Z) \]

Affected Positions

\[ T[3], P[1], Q[2] \]
\[ T[2], P[2], Q[1] \]
VADALOG is based on **Warded Rules**

A Datalog± program is **warded** if for each rule body:

- all dangerous variables jointly occur in a single „ward“ atom, and
- this ward shares only *unaffected* variables with the other body-atoms

\[
\begin{align*}
P(X,Y) \land S(Y,Z) & \rightarrow \exists W \ T(Y,X,W) \\
T(X,Y,Z) & \rightarrow \exists W \ P(W,Z) \\
P(X,Y) & \rightarrow \exists Z \ Q(X,Z)
\end{align*}
\]

Core Vadalog = warded Datalog[∃,⊥,¬-strat]

Clearly, Datalog is contained in Vadalog
Examples of Warded Datalog± Rules

1. Symmetry rule for marriage intervals:

\[
\text{spouse1}(x,y1) \land \text{spouse2}(x,y2) \land \\
\text{start}(x,y3) \land \text{end}(x,y4) \rightarrow \\
\exists v. \text{spouse2}(v,y1) \land \text{spouse1}(v,y2) \land \\
\text{start}(v,y3) \land \text{end}(v,y4)
\]

2. OWL 2 QL description logic:

<table>
<thead>
<tr>
<th>DL-Lite TBox</th>
<th>First-Order Representation (Datalog±)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DL-Lite_{core}</td>
<td>( \forall X \text{ professor}(X) \rightarrow \exists Y \text{ teachesTo}(X,Y) )</td>
</tr>
<tr>
<td>professor (\subseteq) (\exists)teachesTo</td>
<td>( \forall X \text{ professor}(X) \land \neg \text{ student}(X) \rightarrow \bot )</td>
</tr>
<tr>
<td>professor (\subseteq) \neg student</td>
<td></td>
</tr>
<tr>
<td>DL-Lite_{R} (OWL 2 QL)</td>
<td>( \forall X\forall Y \text{ hasTutor}(X,Y) \rightarrow \text{ teachesTo}(Y,X) )</td>
</tr>
</tbody>
</table>
Language Features (selection)

Data types and associated operations & expressions: integer, float, string, Boolean, date, sets.

Monotonic aggregations: min, max, sum, prod, count work even in presence of recursion while preserving monotonicity of set-containment

Example: Company Control

\[\text{own}(x,y,w), \ w>0.5 \rightarrow \text{control}(x,y);\]
\[\text{control}(x,y), \text{own}(y,z,w), \]
\[v=\text{msum}(w, \langle y \rangle), \ v>0.5 \rightarrow \text{control}(x,z).\]

Probabilistic reasoning: facts and rules can be adorned with probabilities. Marginal probabilities for derived facts will be computed assuming independence.

Equality (EGDs, functional dependencies) if non-conflicting.
@bind("Own", "rdbms", "companies.ownerships").

@qbind("Own", "graphDB", "MATCH (a)-[o:Owns]->(b) RETURN a,b,o.weight").

Cypher query (Neo4j)

@bind("q","data source", "schema","table").
@update("q",{1,3,4,5}).
We are currently experimenting with different tools and different types of interfaces and interactions.
Interaction Model 1

- We prepare a relation as ML input.
- ML sw classifies facts and sends them into the core reasoning system.

Interaction Model 2

- ML package acts as a special predicate.
- Called by the core reasoning system.

Interaction Model 3

- ML sw learns rules.
- Rules are translated into probabilistic Vadalog rules.
Core Algorithms

\[ D = \{P(a), Q(a, c)\} \]

1 : \( P(x) \rightarrow \exists z \ Q(x, z) \)
2 : \( Q(x, y) \rightarrow S(y, x) \)
3 : \( S(x, y), P(y) \rightarrow T(y, x) \)
4 : \( T(x, y), Q(z, y) \rightarrow H(x, z) \)
5 : \( T(x, y) \rightarrow Q(x, x) \)
6 : \( Q(x, y) \rightarrow E(y) \)
7 : \( H(x, x) \rightarrow \exists z \ Q(x, z) \)
8 : \( P(x) \rightarrow \exists z \ T(x, z) \).

- Bottom-up chase processing with "aggressive" termination strategy
- Top-down query processing (currently under implementation)
- Advanced program rewriting and optimization techniques
- Efficient & highly scalable cache management, query plan optimization
- Recent evaluation shows the system is extremely competitive
Performance

(Person with significant control over a company)
Crucial Question

We have a powerful language and system for reasoning with rules Over „Big Data“ and can interact with machine learning.

But are there actually real problems that can be solved with a reasonable number of rules?
Crucial Question

We have a powerful language and system for reasoning with rules over „Big Data“ and can interact with machine learning.

*But are there actually real problems that can be solved with a reasonable number of rules?*

*Yes, there are many!, for example:*

- Banks: Fraud detection, [current project]
- Banks: Creditworthiness
- Logistics: Supply chain risks
- Security companies: Detection of critical events
- Fully automated Web data extraction: The DIADEM project
- ... and 10000 more.
DIADEM

Application domain with thousands of websites

BLACKBOX

URL

Application-relevant Structured data (XML or RDF)
machine learning
e.g. neural networks

page classification
visual clue recognition
text classification
small entity identification

rule-based reasoning
knowledge-based
page exploration strategy
navigation planning
global decision making
plausibility checks

Taught knowledge
(expert system)

Self-learned knowledge

Z \leftarrow X \& Y
Rough Idea: Knowledge via Rules

Use “expert” rules that analyze Web pages and interact with them

- **Ontological rules** (how do entities relate to each other)
  - a flat is a real-estate property
  - a house is a real-estate property
  - a real-estate property has a number of rooms
  - a price consists of a number and a currency

- **Phenomenological rules** (how do entities manifest themselves on the Web?)
  - the text chunk closest to an input field is with high prob. its descriptor.
  - each sales item is described in a “convex” (usual. rectangular) region.

- **Site exploration rules:**
  - before filling a field try to leave it empty
  - rules for handling next-page links

- **Other types of rules**
Evaluation on 10k+ Sites

- **10,493 Sites** from real-estate and used-car
- **45 Node** Amazon EC2 cluster running 2.1 days
- **92%** Effective wrappers for more than 92% of sites on average
- **97%** Precision of extracted primary attributes
- **100** Domain-dependent concepts and relations
- **20 Days** (one expert) to adjust system to a new domain
Domains considered so far (since 2014)

- Real estate UK
- Real estate US
- Used cars
- Consumer electronics
- Restaurant chains
- Restaurants in the ‘Open Web’
- Jobs (from company Web sites)
- News
- Companies
Commercial Impact

ERC Advanced Grant DIADEM + ERC Proof of Concept Grant EXTRALYTICS

Wrapidity

Founded February 2015 operating initially in Oxford now in London

Two possibilities:

• Build up company with large client portfolio

• Sell technology, IP & software to strategic partner
Meltwater Acquires Wrapidity to Add AI Capabilities into Media Intelligence Platform

By Sudipto Ghosh

Posted on February 21, 2017

Meltwater, the leading B2B data analytics company, has acquired London-based web data extraction company Wrapidity for an undisclosed amount. The AI-startup that spun out of Oxford University in 2015 will be a separate entity in Meltwater’s existing platform. By beefing up its “media intelligence” platform, Meltwater will now offer AI-powered automation tools for data analytics and media monitoring from unstructured web-based content.

In the era of specialized AI for MarTech, Wrapidity offers tailor-made solutions to content-specific problems arising in image recognition, Natural Language Processing, and machine learning. By acquiring Wrapidity, Meltwater will be able to automate its data extraction processes to reach out to a wide range of online customers based on accurate analytics of historical and real-time data. Meltwater is expected to further improve Wrapidity’s AI capabilities for content discovery and data asset management, enabling marketers to interrogate data for diverse purposes, including sales enablement, social media monitoring and so on.
REFERENCES

KNOWLEDGE GRAPHS & THE VADALOG SYSTEM:


CORE VADALOG, SPARQL, Warded Rules, Complexity, Theory:
Gerald Berger, Georg Gottlob, Andreas Pieris, Emanuel Sallinger: The Space-Efficient Core of Vadalog. PODS 2019: 270-284


DATALOG+-:


DIADEM WEB DATA EXTRACTION SYSTEM: