Knowledge Graphs and Enterprise AI

The Promise of an Enabling Technology

Georg Gottlob

Univ. of Oxford, TU Wien

& DeepReason.ai

VADALOG

Knowledge Graphs and Enterprise AI

The Promise of an Enabling Technology

Georg Gottlob

Univ. of Oxford, TU Wien

& DeepReason.ai

Edinburgh

VADA

EPSRC

PROJECT

Manchester

Oxford

VADALOG

DeepReason.ai

Knowledge Graphs as Large "World" KBs

Cyc [Lenat &Guha 1989]
W: "comprehensive ontology and knowledge base of everyday common sense knowledge".

Freebase [Bollacker et al. 2007] W: "online collection of structured data harvested from many sources, including user-submitted wiki contributions".

Google Knowledge Graph **[**Singhal 2012] **+** **K.Vault** [Dong et al. 2014]
W: "KB used by Google to enhance its search engine's search results with semantic-search information gathered from a wide variety of sources".

DBpedia [Auer et al. 2007]. **Yago** [Suchanek et al 2007]
both generate structured ontologies from Wikipedia.

Wikidata [Vrandečić 2012, Krötzsch+V. 2014] open knowledge base that can be read and edited by both humans and machines.

More Specialized Knowledge Graphs

Facebook Knowledge Graph: Social graph with people, places and things + information from Wikipedia

Amazon Knowledge Graph: Started as product categorization ontology

Wolfram KB: World facts + mathematics

Factual: Businesses & places

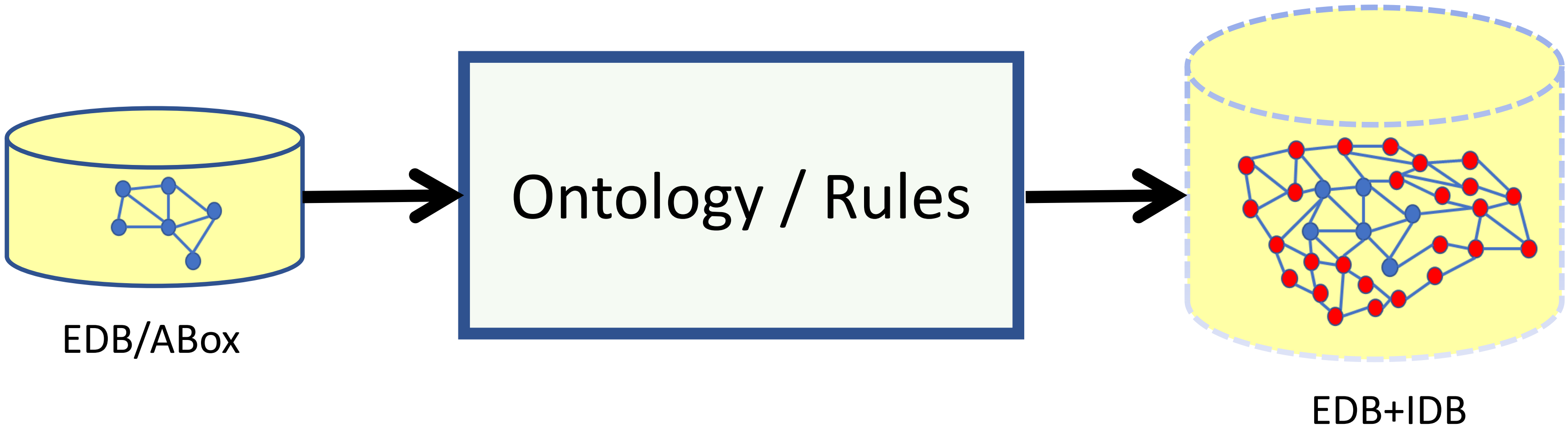
Megagon (Recruit Inst.): People, skills, recruiting

Central Banks: Company register – ownership graph

Credit Rating Agencies …

Thousands of medium to large size companies now want their own corporate knowledge graph. This not just for semantic indexing and search, but for advanced reasoning tasks on top of machine learning.

Reasoning in Knowledge Graphs



Many still think that DLs or graph databases suffice. However:

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)
```

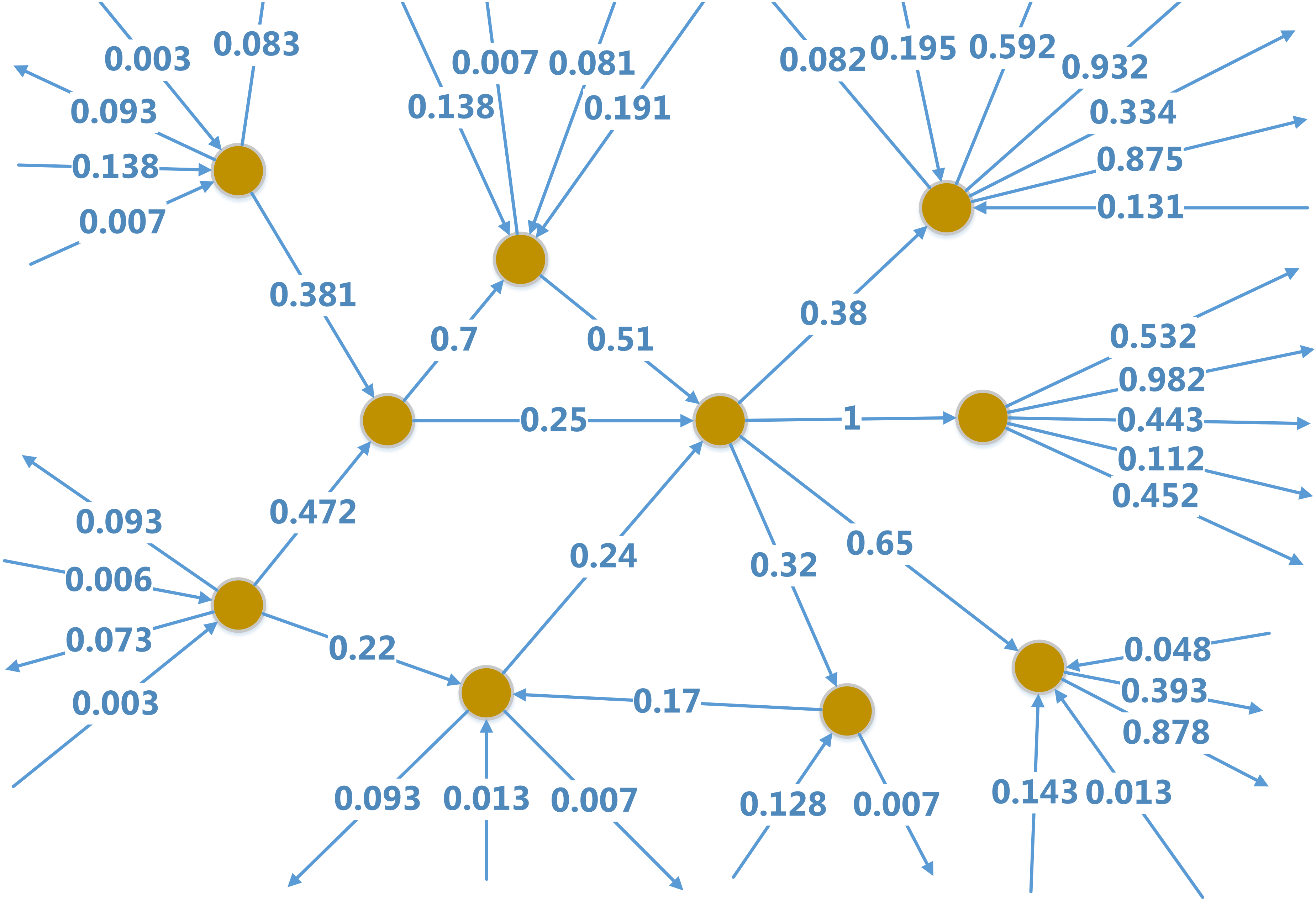
Symmetry rule for marriage intervals in Datalog:

$$\forall\, u,v,x,y.\ \mathtt{married}(u,v,x,y) \rightarrow \mathtt{married}(v,u,x,y)$$

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

```
                                        company(x)  → own(x,x).
                            own(x,y,w), w>0.5 → control(x,y).
control(x,y), own(y,z,w), v=msum(w,⟨y⟩), v>0.5 → control(x,z).
```

This cannot be expressed in DLs and only clumsily in SQL and Graph DBMS!

Example: My Creditworthiness

Example: My Creditworthiness

up to £10,000

£8,500

£12,000

up to EUR 10,000

up to EUR 20,000

£500

£ 8,000

£ 12,500

EUR 14,000

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 <u>wrong data</u>.

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 <u>wrong data</u>.

A human credit rating expert would instead use of the rule:

> ***If property owners move into their recently bought one-family property, then the previous occupiers have <u>most likely</u> moved out.***
>
> ***(Such updates are often missing in the database)***

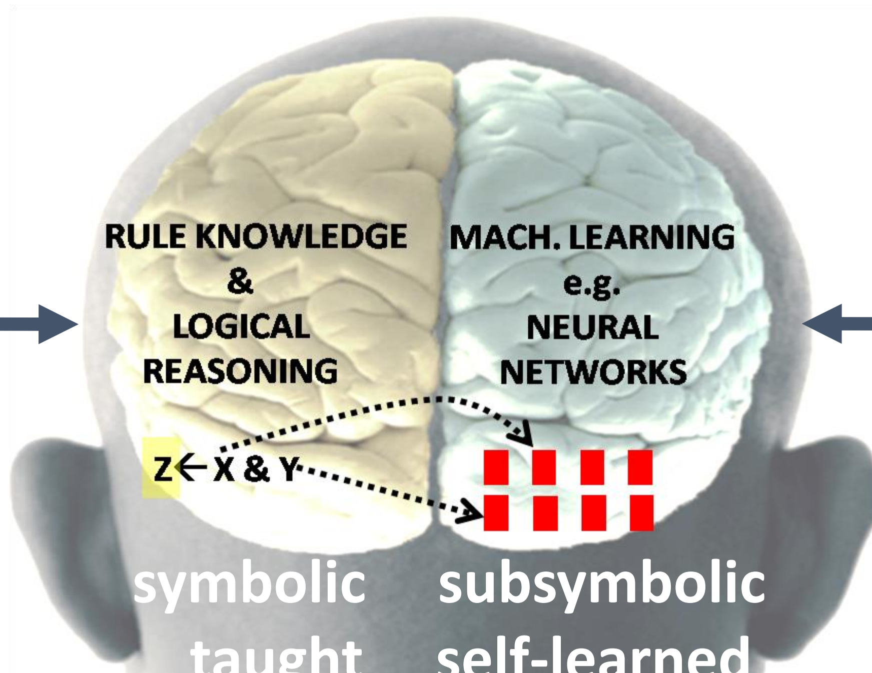
This rule can be used to update the database <u>before</u> applying machine learning.

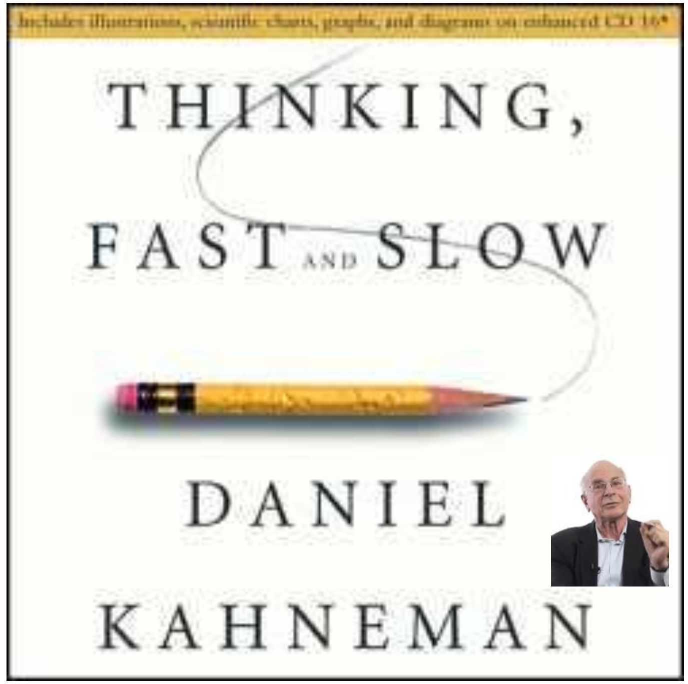
Knowledge Graph Management Systems (KGMS)

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

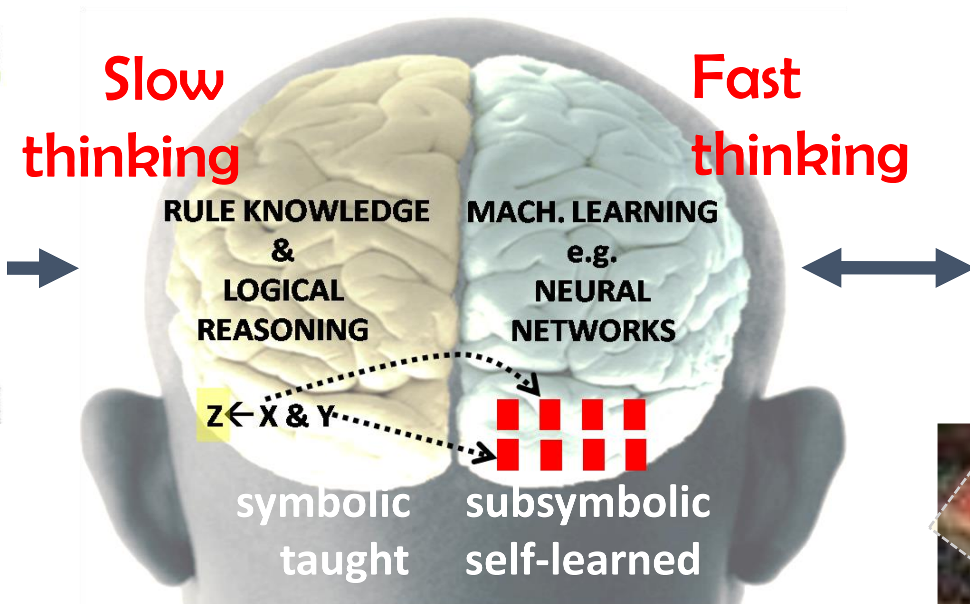
KGMS = KBMS + Big Data + Analytics

Misusing the lateralization thesis for illustration





Grandma: “Fly agarics are poisonous mushrooms.
If you eat a poisonous mushroom, you may die”.



Desiderata for KGMS According to our Philosophy

No extra permanent data repository or database/DBMS

- Uses (possible multiple) <u>existing</u> company data repositories/databases
- Can query and update these – streaming into main memory for reasoning
- No data migration necessary

Multiple data models possible.

- Relational, graph, RDF, …
- Reasoning engine interprets all data relationally (by Datalog facts)

High expressive power of reasoning language; express at least:

- Full Datalog with full recursion and stratified negation
- Graph navigation
- Aggregate functions
- Description logics such as: DL-Lite (OWL 2 QL), EL, F-Logic Lite
- SPARQL under RDFS or OWL 2 QL Entailment Regimes

Good complexity and scalability

- Tractability guarantee for main formalism
- Highly efficient, and highly parallelizable language fragments

Support for machine learning, analytics, and collaborative filtering

- APIs to standard ML and analytics packages (do not reinvent the wheel)
- Provide system support for graph analysis (e.g. balanced separators), and typical functions such as *argmin* (with grad. desc.), *eigenvector*, *pagerank*, *simrank*, etc.

Knowledge Graph Management Systems

a diverse new field – many systems with different capabilities

GRAKN.AI

Graph Path

AllegroGraph
Franz Inc.

Stardog

DeepReason.ai

neo4j

metaphacts

MAANA™

GNOSS

Analysis along many dimensions possible

AllegroGraph Franz Inc.	Graph database supporting SPARQL and Prolog reasoning
GRAKN.AI	Apache Cassandra-based KGMS providing schema support based on the Entity Relationship model
	Knowledge Graph-as-a-Service
VADALOG	Data source-agnostic KGMS supporting ontological and recursive reasoning based on Datalog
neo4j	Leading graph database system
Stardog	RDF-based unifying data-integration platform
metaphacts	SPARQL 1.1-graph database-based end-user-oriented platform
MAANA	Azure-based computation-focused platform
GNOSS	RDF and OWL-based metadata management solution.

Migration necessary?

AllegroGraph Franz Inc.	Graph database supporting SPARQL and Prolog reasoning
GRAKN.AI	Apache Cassandra-based KGMS providing schema support based on the Entity Relationship model
	Knowledge Graph-as-a-Service
VADALOG	Data source-agnostic KGMS supportin recursive reasoning based on Datalog
neo4j	Leading graph database system
Stardog	RDF-based unifying data-integration p
metaphacts	SPARQL 1.1-graph database-based end-user-oriented platform
MAANA	Azure-based computation-focused platform
GNOSS	RDF and OWL-based metadata management solution.

uses existing company DBMS for permanent storage

uses existing company DBMS for permanent storage

Principle Data Format / Backend

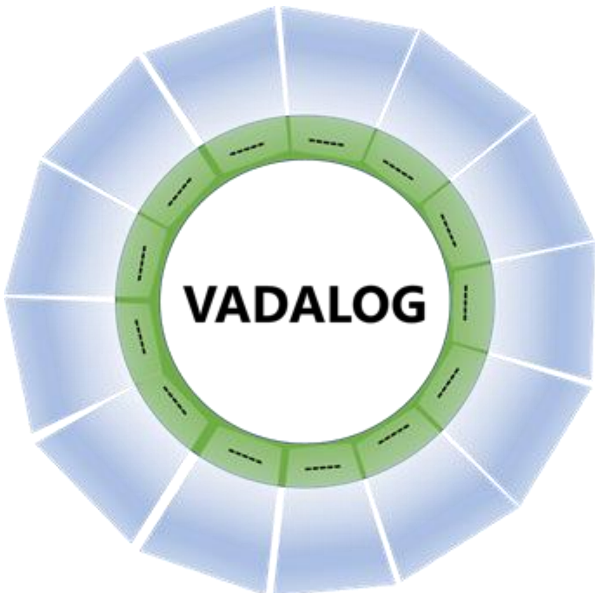
AllegroGraph Franz Inc.	Graph database sup… d Prolog reasoning	Graph
GRAKN.AI	Apache Cassandra-based KGMS providing s… based on the Entity Relationship model	Cassandra
GP	Knowledge Graph-as-a-Service	
VADALOG	Data source-agnostic KGMS supporting ont… recursive reasoning based on Datalog	Multiple
neo4j	Leading graph datab…	Graph
Stardog	F… data-integration platform	RDF
metaphacts	S… database-based end-user-oriented p…	RDF
MAANA	Azure-based computation-focused platform	Azure
GNOSS	I… d metadata management solution.	RDF

Analysis along many dimensions possible

AllegroGraph Franz Inc.	Graph database supporting SPARQL and Prolog reasoning
GRAKN.AI	Apache Cassandra-based KGMS providing schema support based on the Entity Relationship model
GP	Knowledge Graph-as-a-Service
VADALOG	Data source-agn… recursive reason…
neo4j	Leading graph database system
Stardog	RDF-based unifying data-integration platform
metaphacts	SPARQL 1.1-graph database-based end-user-oriented platform
MAANA	Azure-based computation-focused platform
GNOSS	RDF and OWL-based metadata management solution.

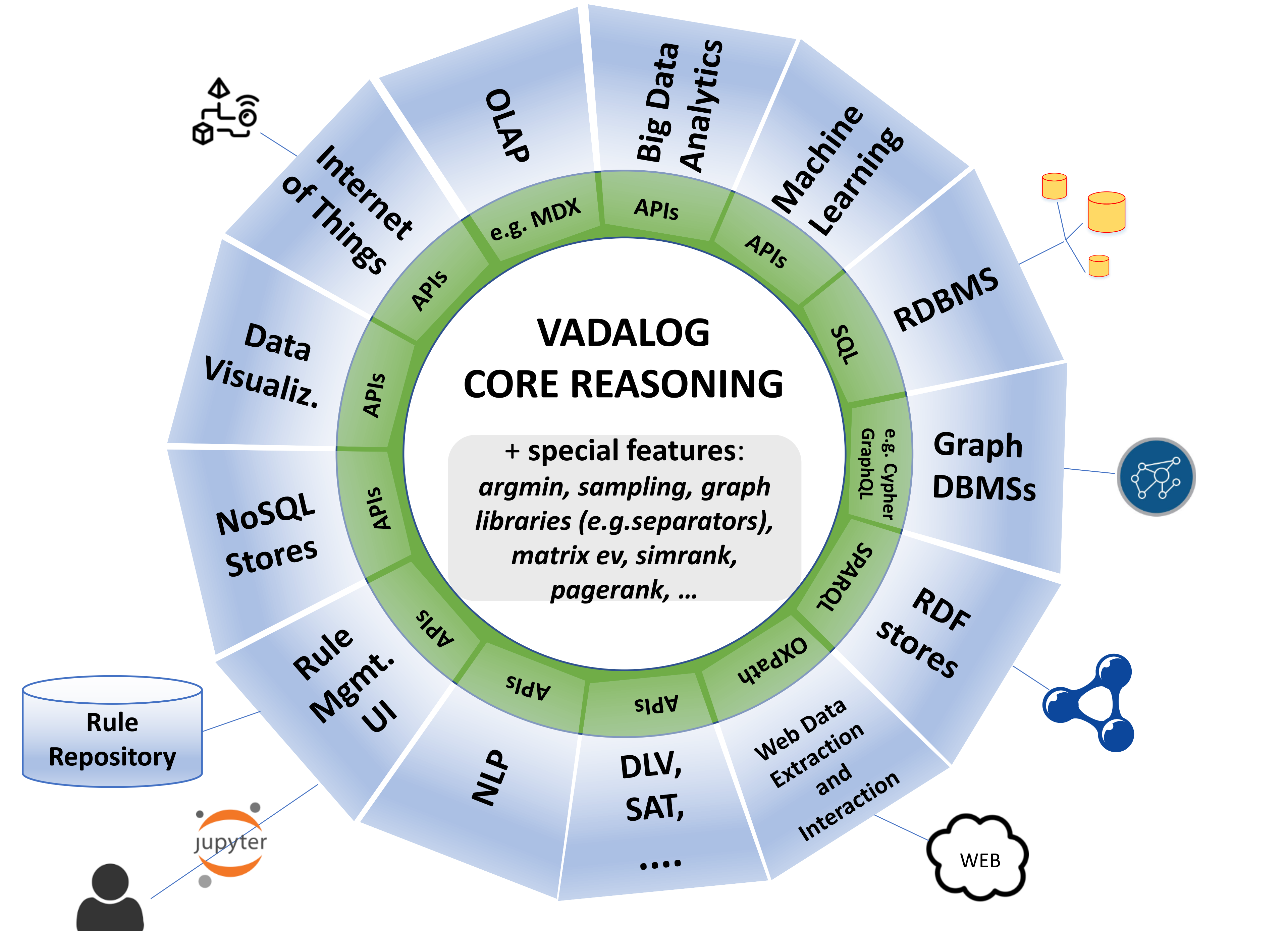
To our best knowledge, the most expressive reasoning language with tractability guarantees

Vadalog KGMS Being Built at Oxford



Current Team Members

- VADA = **V**alue-**A**dded **DA**ta
- General architecture of VADALOG system
- Core reasoning language VADALOG = Warded Datalog + extensions
- Connectivity: Some plug-ins



VADALOG
CORE REASONING
+ special features:
argmin, sampling, graph libraries (e.g.separators), matrix ev, simrank, pagerank, …
OLAP
e.g. MDX
Big Data Analytics
APIs
Machine Learning
APIs
RDBMS
SQL
Graph DBMSs
e.g. Cypher GraphQL
RDF stores
SPARQL
Web Data Extraction and Interaction
OXPath
WEB
DLV, SAT, ….
APIs
NLP
APIs
Rule Mgmt. UI
APIs
Rule Repository
jupyter
NoSQL Stores
APIs
Data Visualiz.
APIs
Internet of Things
APIs

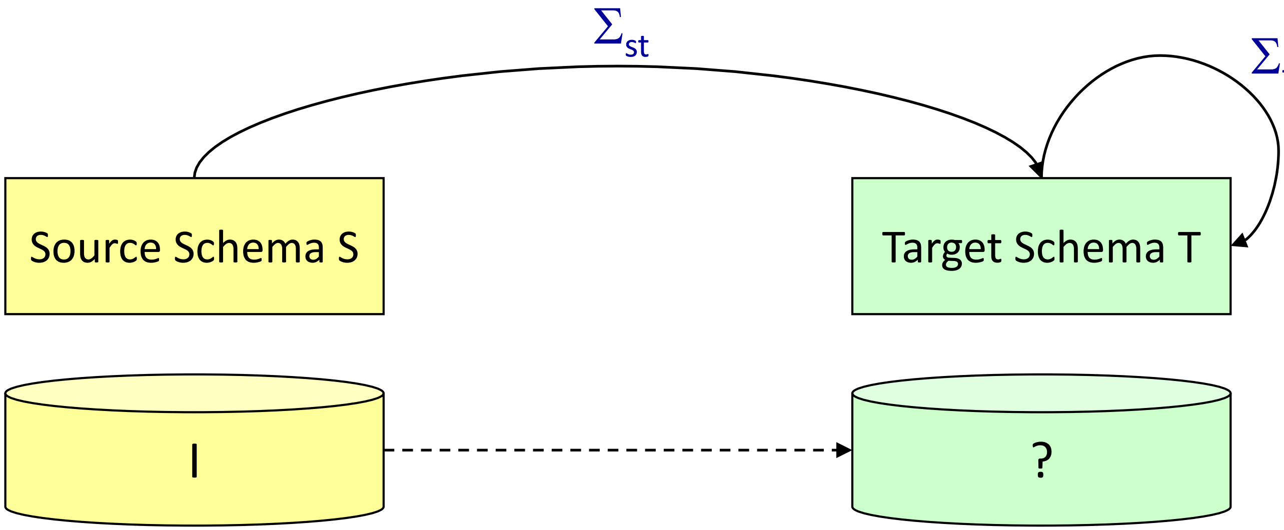
Vadalog: The Core Reasoning Language

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

Why existential quantifiers in rule heads?

- Data exchange, data integration
- Data extraction
- Reasoning with RDF → Wikidata example
- Ontology querying (DL-Lite, EL, etc.)
- Data anonymization
- Duplicate handling
- Automated product configuration
- Conceptual Modeling (e.g., UML)

Data Exchange, Data Provisioning, Data Wrangling



employee(*Lastname*, *Firstname*, *Address*)

person(*FirstName*, *Lastname*, *Birthdate*)

$$\texttt{employee(X,Y,Z)} \rightarrow \exists \texttt{W}\ \texttt{person(Y,X,W)}$$

[Fagin, Kolaitis, Miller & Popa, 2003]; [Arenas et al., 2014]

Object Creation
e.g. in web data extraction

PRODUCT	PRICE
Toshiba_Protege_cx	480
Dell_25416	360
Dell_23233	470
Acer_78987	390

Object Creation
e.g. in web data extraction

T_1	T_2
PRODUCT	PRICE
Toshiba_Protege_cx	480
Dell_25416	360
Dell_23233	470
Acer_78987	390

Object Creation
e.g. in web data extraction

T_1	T_2
PRODUCT	PRICE
Toshiba_Protege_cx	480
Dell_25416	360
Dell_23233	470
Acer_78987	390

Object Creation
e.g. in web data extraction

```
table(T₁),
table(T₂),
sameColor(T₁,T₂),
isNeighbourRight(T₁,T₂) →
                    ∃T tablebox(T),
                       contains(T,T₁),
                       contains(T,T₂).
```

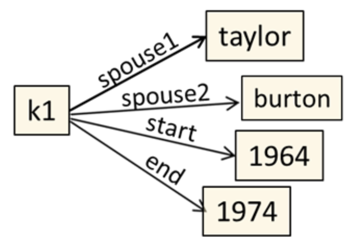
T_1	T_2
PRODUCT	PRICE
Toshiba_Protege_cx	480
Dell_25416	360
Dell_23233	470
Acer_78987	390

Reasoning with RDF – Foreign Key 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):

```
spouse1(k1,taylor),
spouse2(k1,burton),
start(k1,1964),
end(k1,1974)
```

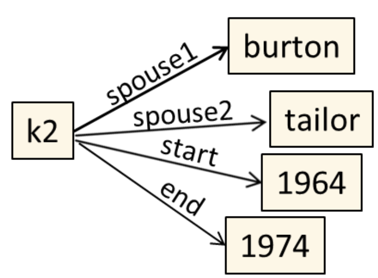



```
∀ u,v,x,y. married(u,v,x,y) → married(v,u,x,y)
```

This symmetry rule for marriage intervals now becomes:

```
spouse1(u,y1) ∧ spouse2(u,y2) ∧ start(u,y3) ∧ end(u,y4) →
∃v.spouse(v,y1) ∧ spouse1(v,y2) ∧ start(v,y3) ∧ end(v,y4)
```

```
spouse1(k2,burton),
spouse2(k2,taylor),
start(k2,1964),
end(k2,1974)
```



Description Logics & Ontological Reasoning

The DL-Lite Family

Popular family of DLs with low (AC_0) data complexity

DL-Lite TBox	First-Order Representation (Datalog$^\pm$)
DL-Lite$_{core}$	
$\mathit{professor} \sqsubseteq \exists \mathit{teachesTo}$	$\forall X\ \mathit{professor}(X) \rightarrow \exists Y\ \mathit{teachesTo}(X,Y)$
$\mathit{professor} \sqsubseteq \neg \mathit{student}$	$\forall X\ \mathit{professor}(X) \wedge \mathit{student}(X) \rightarrow \bot$
DL-Lite$_R$ (OWL 2 QL)	
$\mathit{hasTutor}^- \sqsubseteq \mathit{teachesTo}$	$\forall X \forall Y\ \mathit{hasTutor}(X,Y) \rightarrow \mathit{teachesTo}(Y,X)$
DL-Lite$_F$	
$\mathrm{funct}(\mathit{hasTutor})$	$\forall X \forall Y \forall Z\ \mathit{hasTutor}(X,Y) \wedge \mathit{hasTutor}(X,Z) \rightarrow Y = Z$

[Calvanese, De Giacomo, Lembo, Lenzerini & Rosati, J. Autom. Reasoning 2007]

Datalog[∃]: Full Datalog augmented with ∃-quantifier

Unfortunately:

Theorem: Reasoning ($KB \vDash q$) with Datalog[∃] is undecidable.

[Beeri & Vardi, 1981]; [J. Mitchell 1983] [Chandra & Vardi 1985];

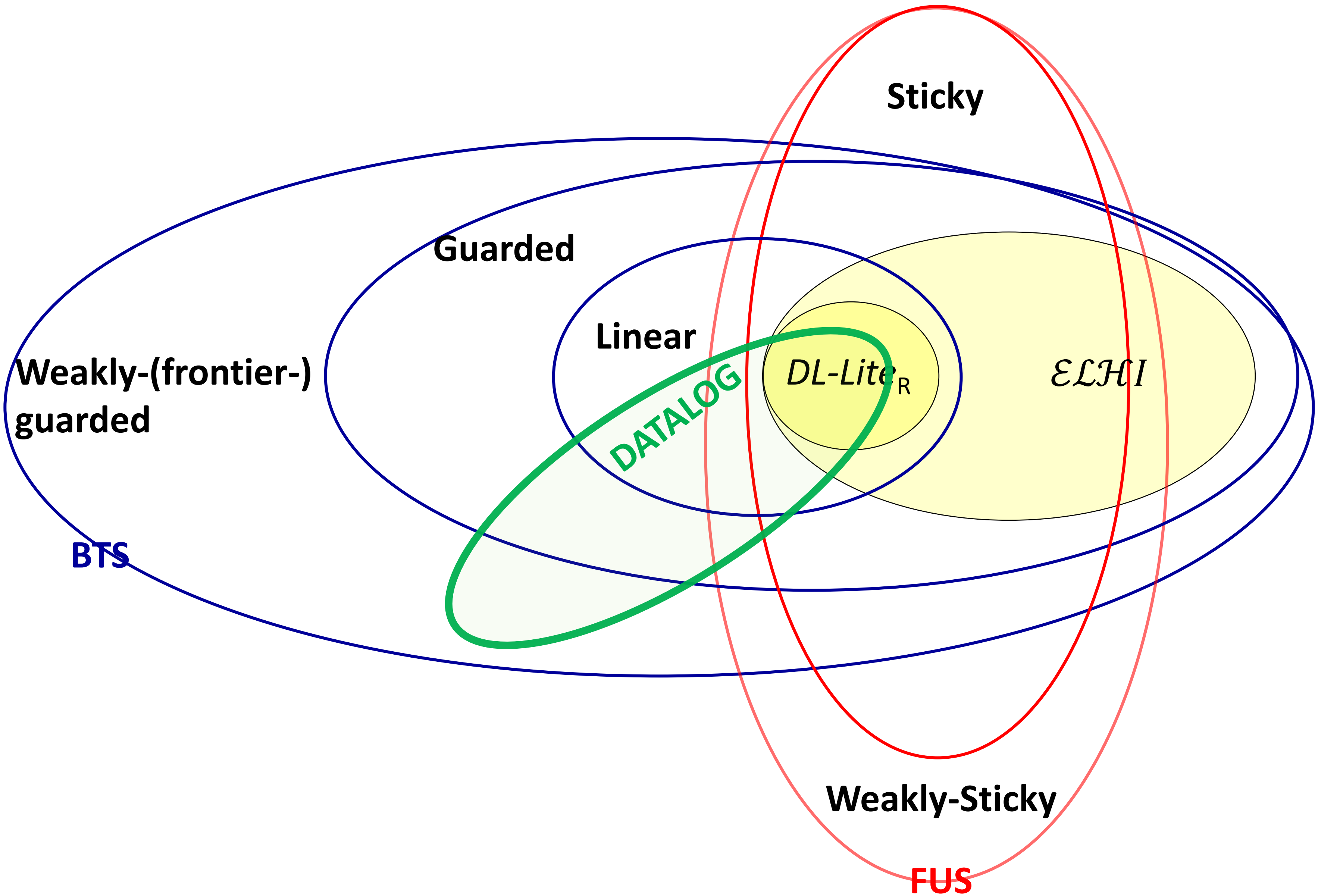
[Calì, G., & Kifer, 2008]; [Baget, Leclère & Mugnier, 2010]

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

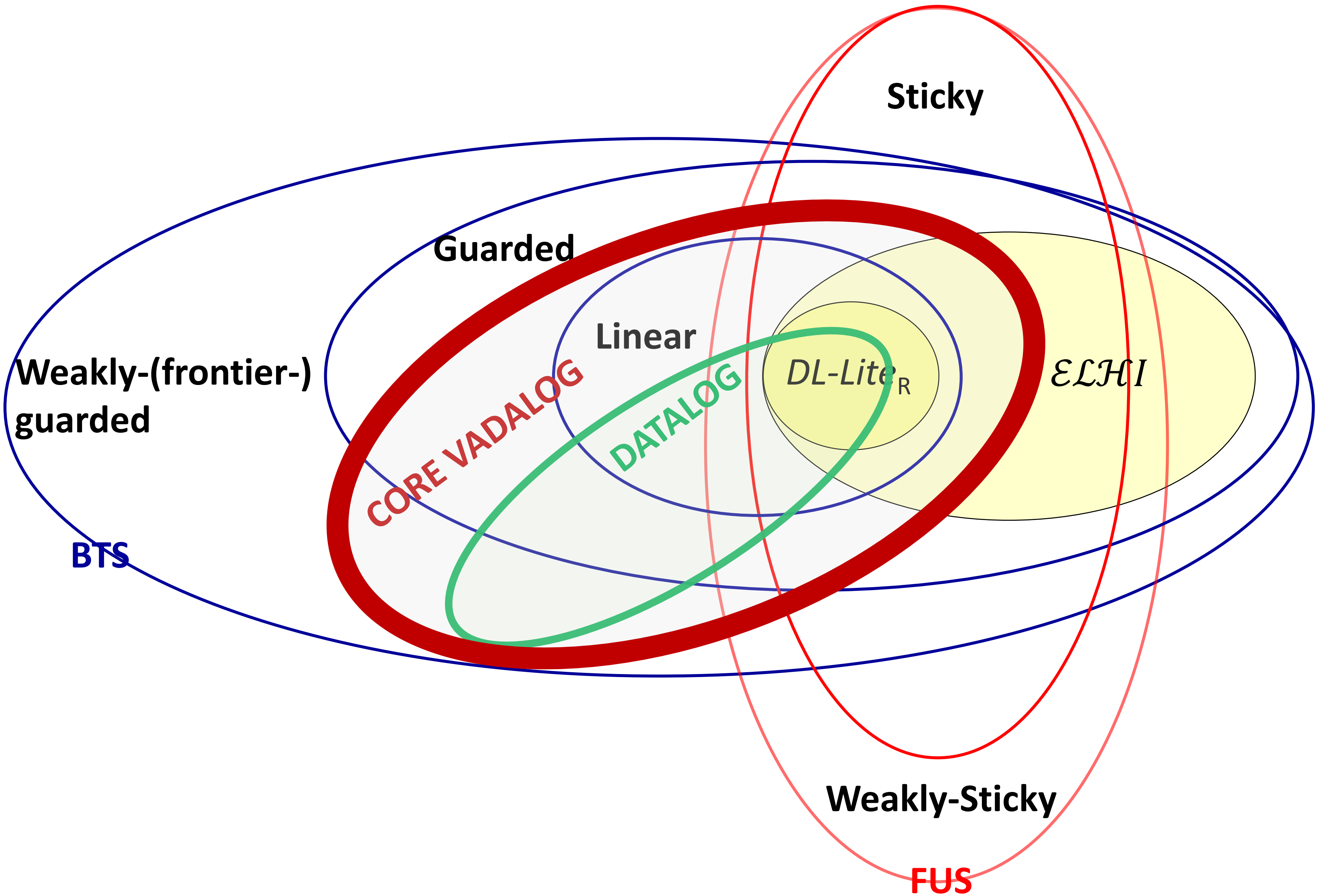
Datalog$^{\pm}$: Datalog[∃,⊥,¬strat, ...] subject to syntactic restrictions.

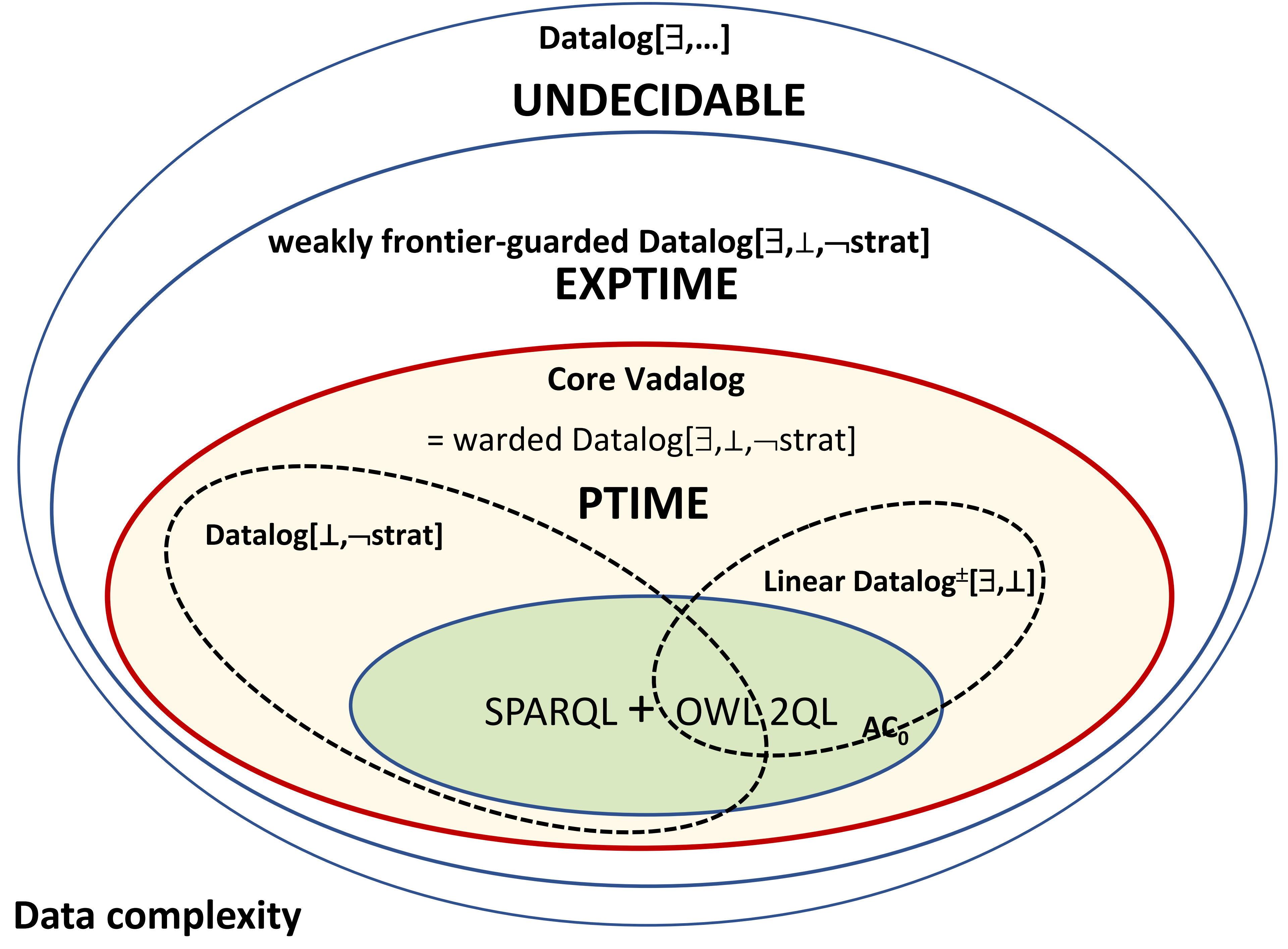
Vadalog: member of the Datalog$^{\pm}$ family admitting efficient reasoning methods.

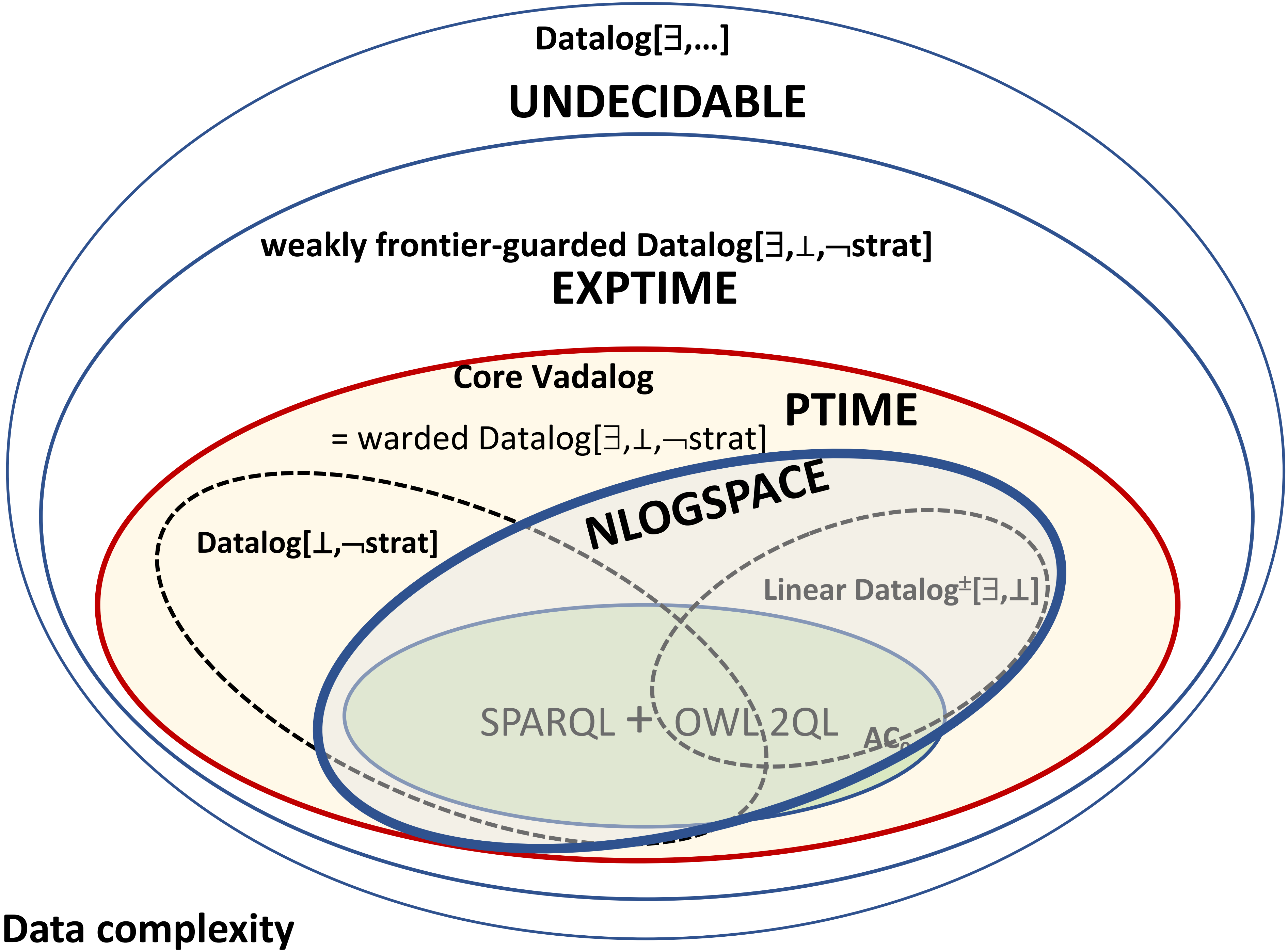
Main Decidable Datalog$^{\pm}$ Languages



Main Decidable Datalog$^{\pm}$ Languages




Datalog[∃,...]
UNDECIDABLE
weakly frontier-guarded Datalog[∃,⊥,¬strat]
EXPTIME
Core Vadalog
= warded Datalog[∃,⊥,¬strat]
PTIME
Datalog[⊥,¬strat]
Linear Datalog±[∃,⊥]
SPARQL + OWL 2QL
AC0
Data complexity



Datalog[∃,...]
UNDECIDABLE
weakly frontier-guarded Datalog[∃,⊥,¬strat]
EXPTIME
Core Vadalog
PTIME
= warded Datalog[∃,⊥,¬strat]
NLOGSPACE
Datalog[⊥,¬strat]
Linear Datalog±[∃,⊥]
SPARQL + OWL 2QL
AC0
Data complexity

Vadalog is based on **Warded Rules**

A Datalog$^{\pm}$ 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

$$P(\boxed{\underline{X}},\underline{Y}),\ S(Y,Z) \rightarrow \exists W\ T(Y,\underline{X},\underline{W})$$

$$T(X,\underline{Y},\boxed{\underline{Z}}) \rightarrow \exists W\ P(\underline{W},\underline{Z})$$

$$P(\boxed{\underline{X}},\underline{Y}) \rightarrow \exists Z\ Q(\underline{X},\underline{Z})$$

Affected Positions

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

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

Core Vadalog = warded Datalog[$\exists,\perp,\neg$strat]

Examples of Warded Datalog$^{\pm}$ Rules

1. **Symmetry rule for marriage intervals:**

$$\texttt{spouse1}(\underline{\texttt{x}},\texttt{y1}) \wedge \texttt{spouse2}(\underline{\texttt{x}},\texttt{y2}) \wedge \texttt{start}(\underline{\texttt{x}},\texttt{y3}) \wedge \texttt{end}(\underline{\texttt{x}},\texttt{y4}) \rightarrow \exists \texttt{v}.\ \texttt{spouse2}(\underline{\texttt{v}},\texttt{y1}) \wedge \texttt{spouse1}(\underline{\texttt{v}},\texttt{y2}) \wedge \texttt{start}(\underline{\texttt{v}},\texttt{y3}) \wedge \texttt{end}(\underline{\texttt{v}},\texttt{y4})$$

2. : **OWL 2 QL description logic**

DL-Lite TBox	Representation in Vadalog
DL-Lite$_{core}$	
$professor \sqsubseteq \exists teachesTo$	$\forall X\ professor(X) \rightarrow \exists Y\ teachesTo(X,Y)$
$professor \sqsubseteq \neg student$	$\forall X\ professor(X) \wedge student(X) \rightarrow \perp$
DL-Lite$_R$ (OWL 2 QL)	
$hasTutor^{-} \sqsubseteq teachesTo$	$\forall X \forall Y\ hasTutor(X,Y) \rightarrow teachesTo(Y,X)$

Theorem

Vadalog can express:

- Datalog with full recursion and stratified negation
- Description logics: DL-Lite Family, in particular, OWL 2 QL, EL, F-Logic Lite
- SPARQL under RDFS and OWL 2 QL Entailment Regimes

Theorem

Vadalog can express:

- Datalog with full recursion and stratified negation
- Description logics: DL-Lite Family, in particular,
  OWL 2 QL, EL, F-Logic Lite
- SPARQL under RDFS and OWL 2 QL Entailment Regimes

Moreover: All queries of iBench can be expressed in Vadalog!

Further 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

```
own(x,y,w), w>0.5 → control(x,y);
control(x,y),own(y,z,w),
  v=msum(w,⟨y⟩), v>0.5 → control(x,z).
```

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

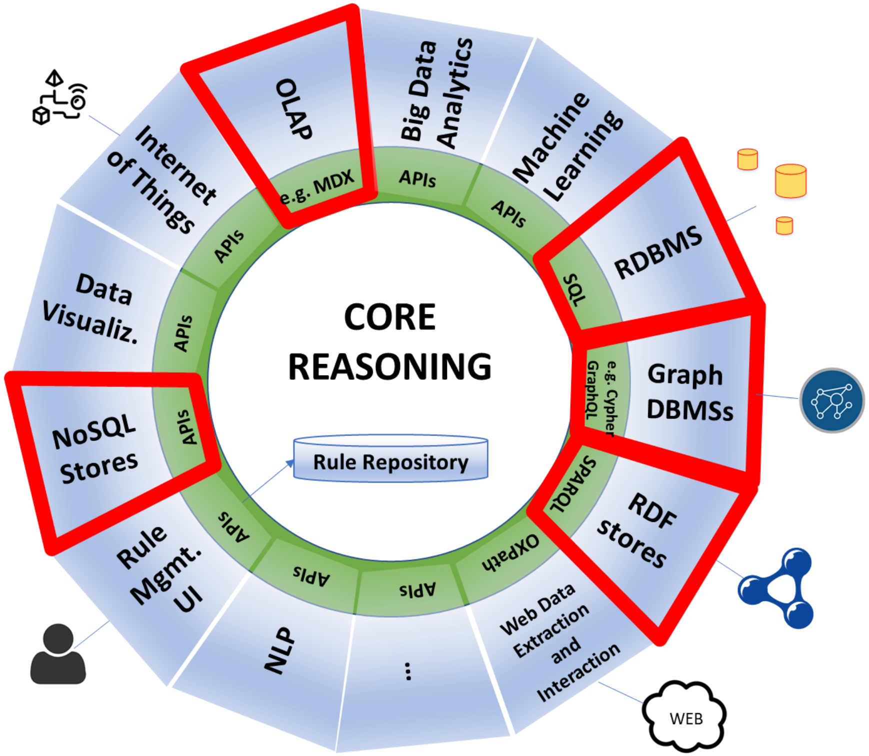
Equality (EGDs, functional dependencies) if non-conflicting.

Rules can be uncertain

```
@weight(0.6)  company(C) → ∃C1 own(C,C1).
@weight(0.5)  own(C,S), holding(C) → subsidiary(S).
```

- A Soft Vadalog rule has a **weight**

- Similar to Markov Logic Network, but Soft Vadalog
  - is not full First Order Logic
  - allows recursive definitions
  - has unrestricted domain



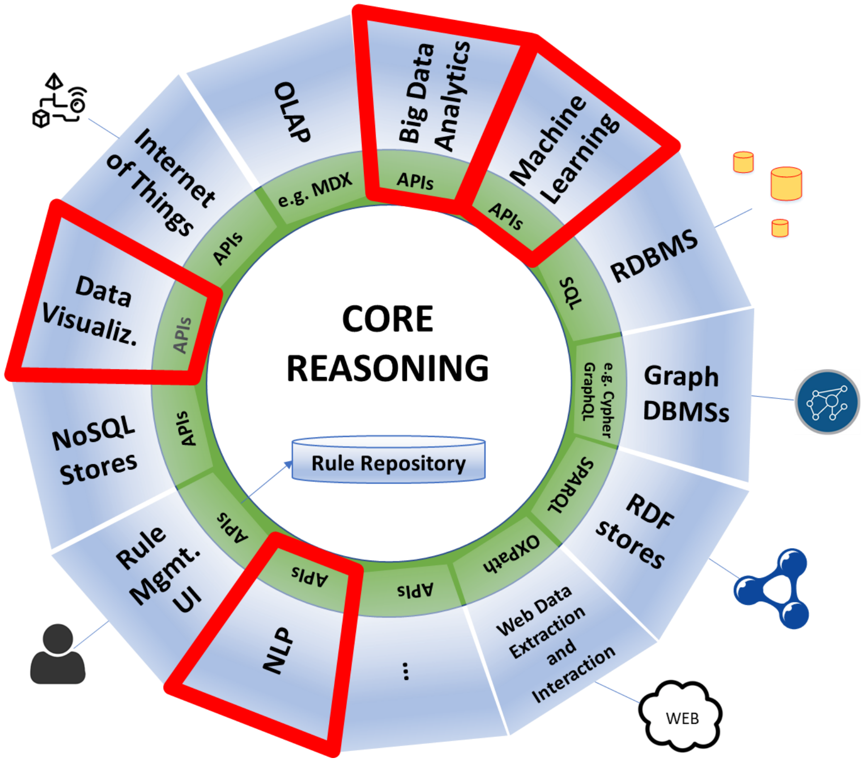
Database Interface

```
@bind("Own", "rdbms", "companies.ownerships").
```

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

Cypher query ([Neo4j](Neo4j))

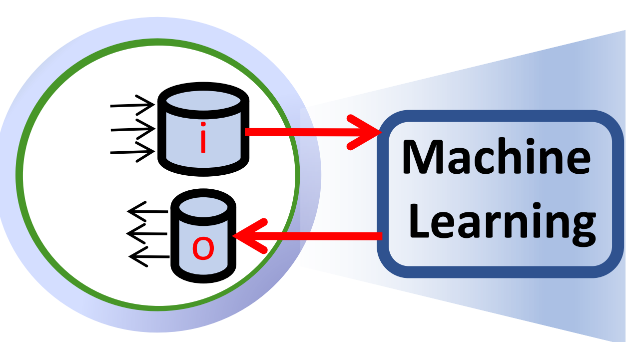
```
@bind("q","data source", "schema","table").
@update("q",{1,3,4,5}).
```



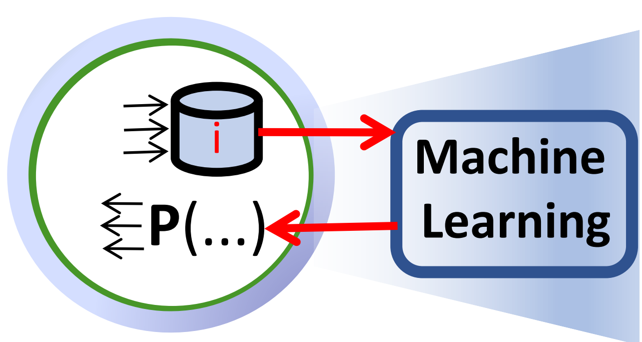
Machine Learning, Big Data Analytics, NLP & Data Visualization

We are currently experimenting with different tools and different types of interfaces and interactions.

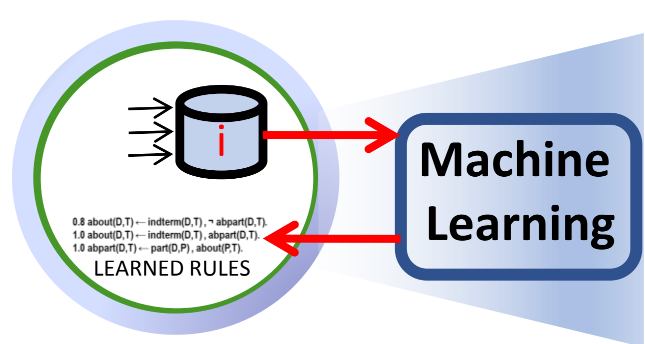
Interaction Model 1

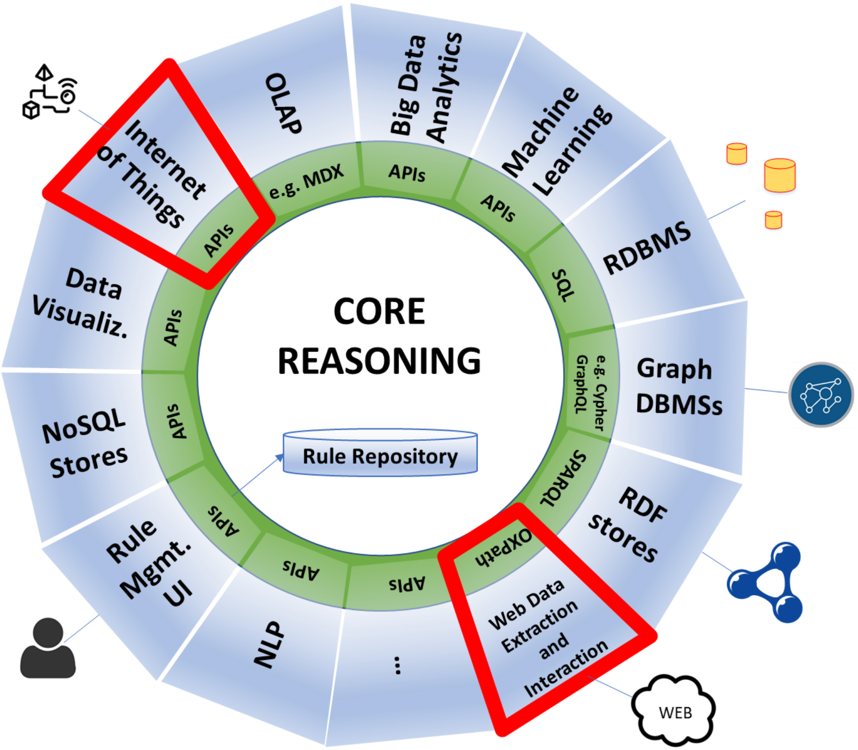


Interaction Model 2



Interaction Model 3





Web Data Extraction & IoT

Interfacing KG to OXPath;
Binding OXPath to Datalog

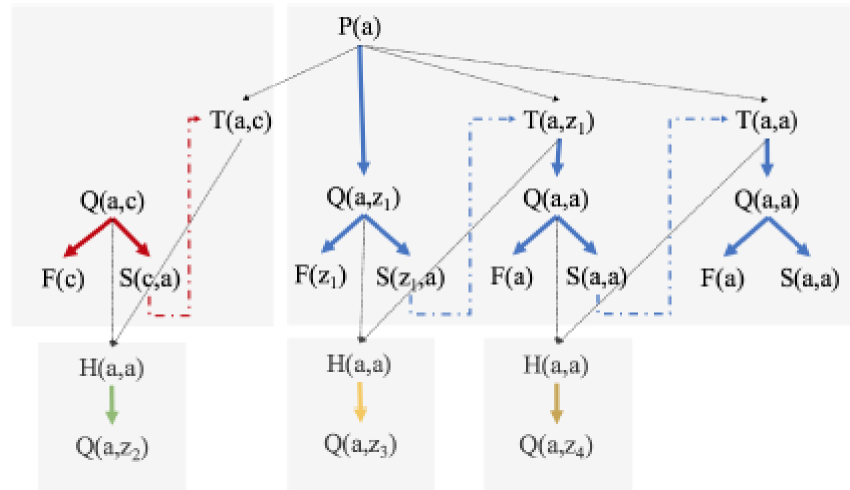
```
@qbind("Own", "oxpath",
            "doc('http://company_register.com/ownerships')
            /descendant::field()[1]/{$1}
            /following::a[.#='Search']/{click/}
            /(//a[.#='Next']/ {click/})*
            //div[@class='c']: [./span[1]:][./span[3]:]")
```

[Furche, T., Gottlob, G., Grasso, G., Schallhart, C., & Sellers, A. (2013).
OXPath: A language for scalable data extraction, automation, and crawling on the deep web.
The VLDB Journal, *22*(1), 47-72. ".]

Core Algorithms

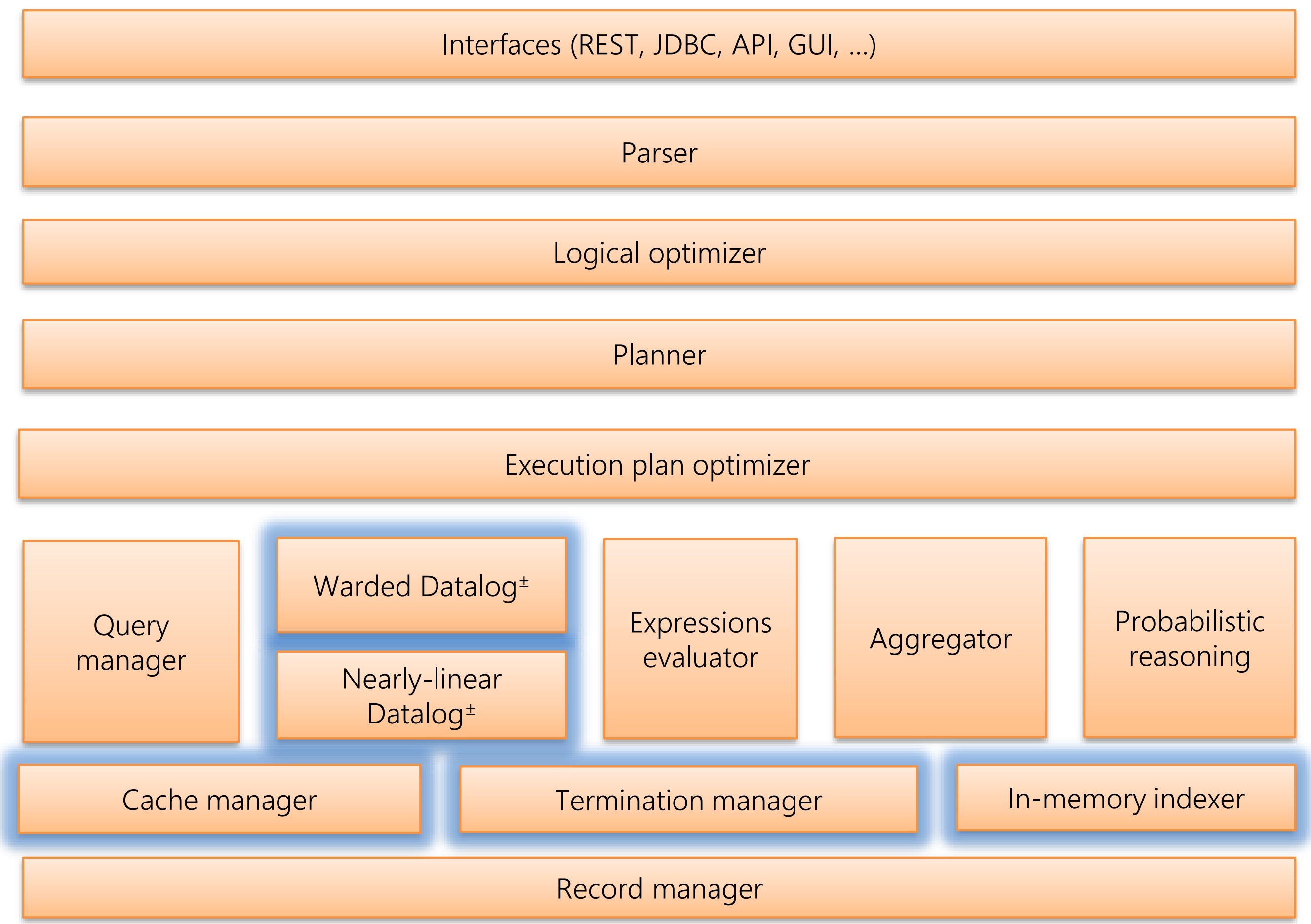
$$D = \{\mathrm{P}(a), \mathrm{Q}(a,c)\}$$

$$1: \mathrm{P}(x) \rightarrow \exists z\, \underline{\mathrm{Q}}(x, \hat{z})$$
$$2: \mathrm{Q}(x, \hat{y}) \rightarrow \mathrm{S}(\hat{y}, x)$$
$$3: \mathrm{S}(\hat{x}, y), \mathrm{P}(y) \rightarrow \mathrm{T}(y, \hat{x})$$
$$4: \mathrm{T}(x, \hat{y}), \mathrm{Q}(z, \hat{y}) \rightarrow \underline{\mathrm{H}}(x, z)$$
$$5: \mathrm{T}(x, \hat{y}) \rightarrow \underline{\mathrm{Q}}(x, x)$$
$$6: \mathrm{Q}(x, \hat{y}) \rightarrow \underline{\mathrm{F}}(\hat{y})$$
$$7: \mathrm{H}(x, x) \rightarrow \exists z\, \underline{\mathrm{Q}}(x, \hat{z})$$
$$8: \mathrm{P}(x) \rightarrow \exists z\, \mathrm{T}(x, \hat{z}).$$



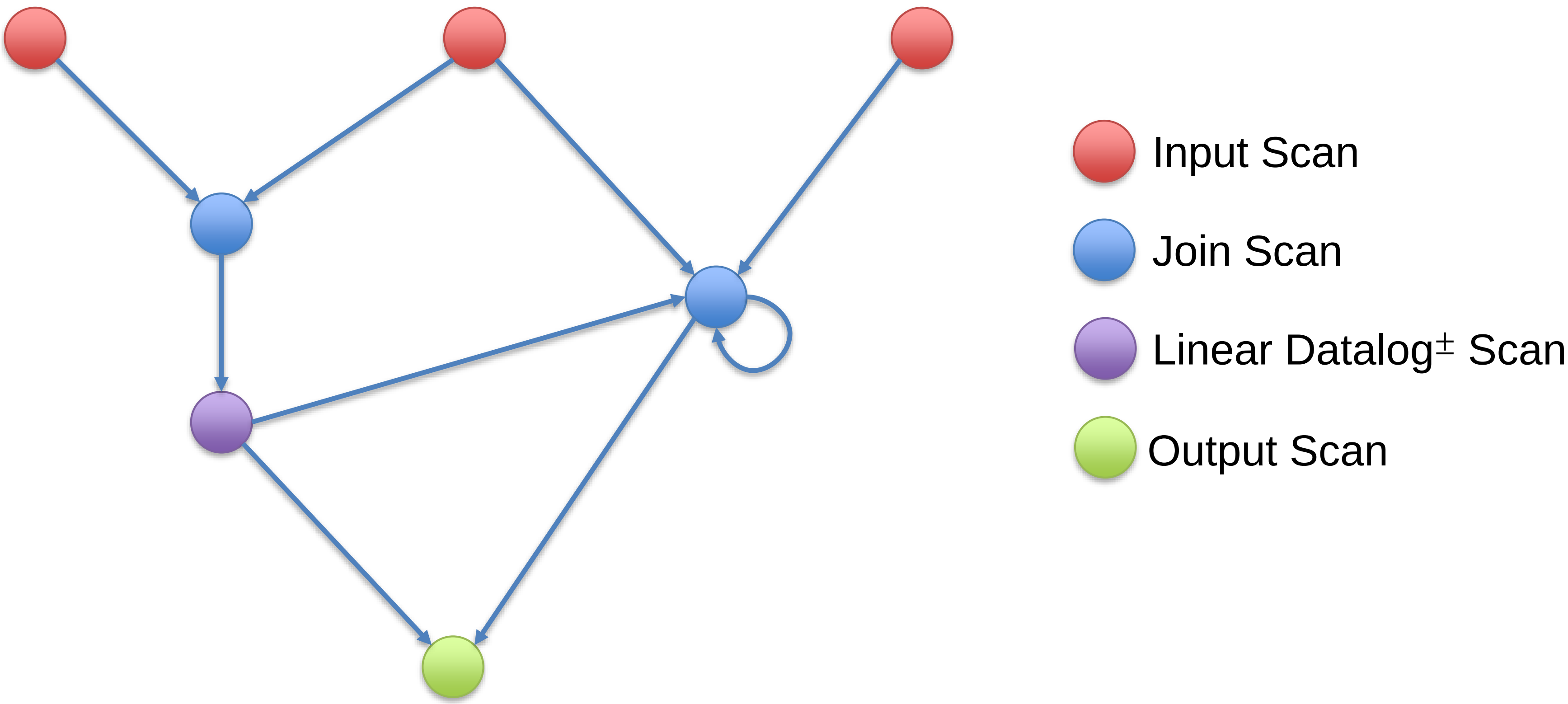
For more details see Luigi Bellomarini, Emanuel Sallinger, Georg Gottlob: The Vadalog System: *Datalog-based Reasoning for Knowledge Graphs*. PVLDB 11(9) 2018

- 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 managmt., query plan optimization
- Recent evaluation shows the system is extremely competitive



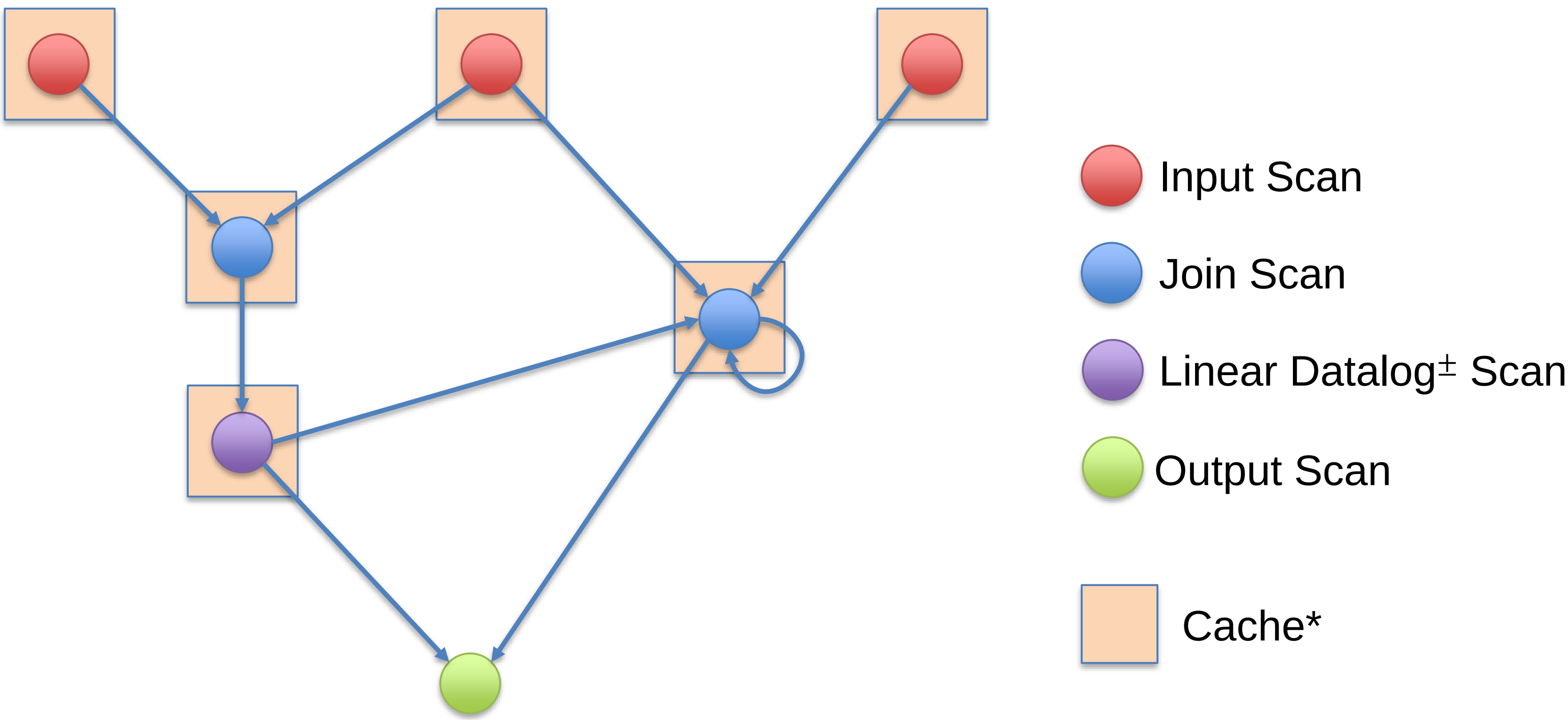
→ PVLDB 11(9) 2018.

In-Memory Stream Processing



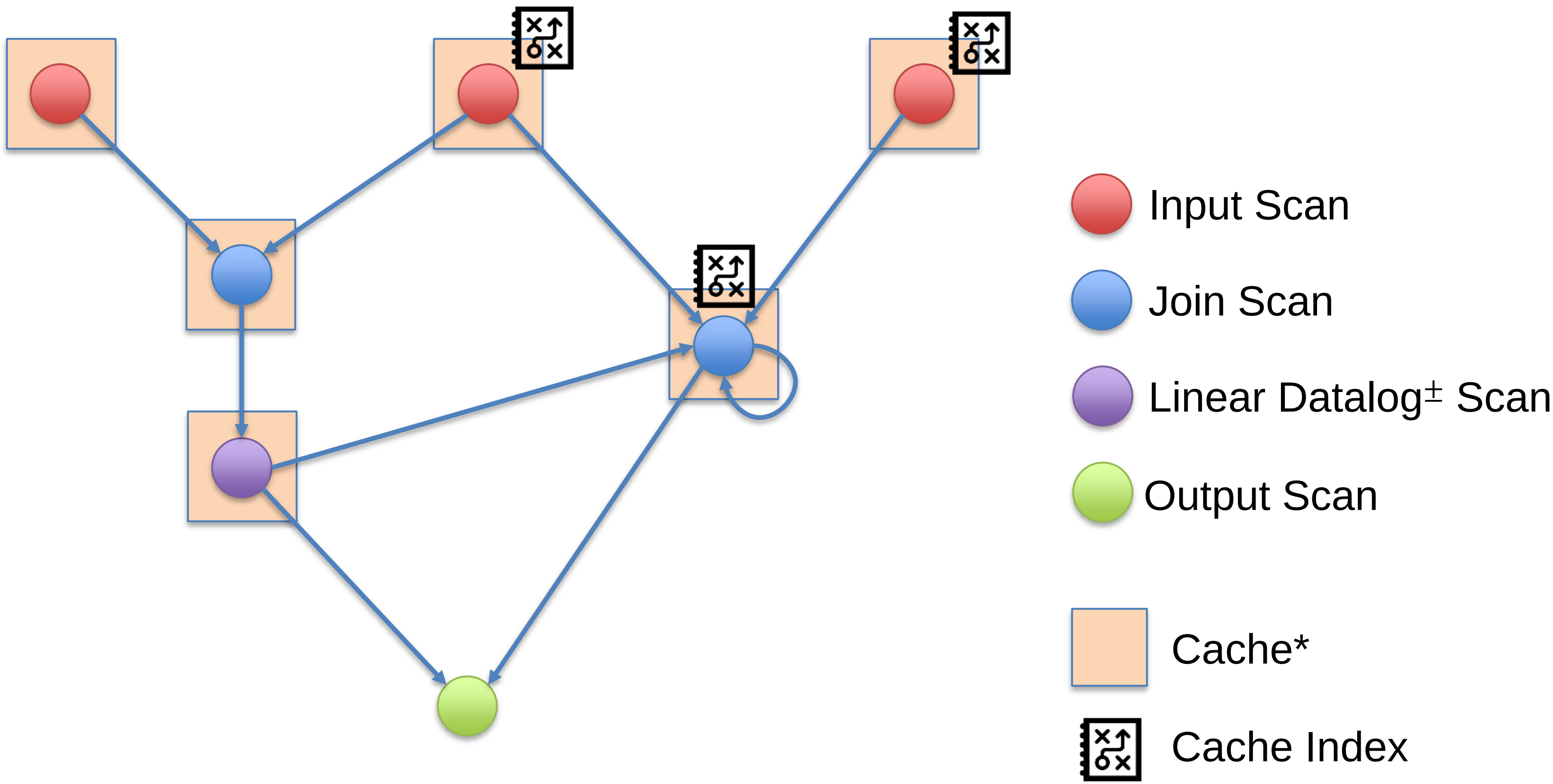
Similar to Volcano iterator model

In-Memory Stream Processing



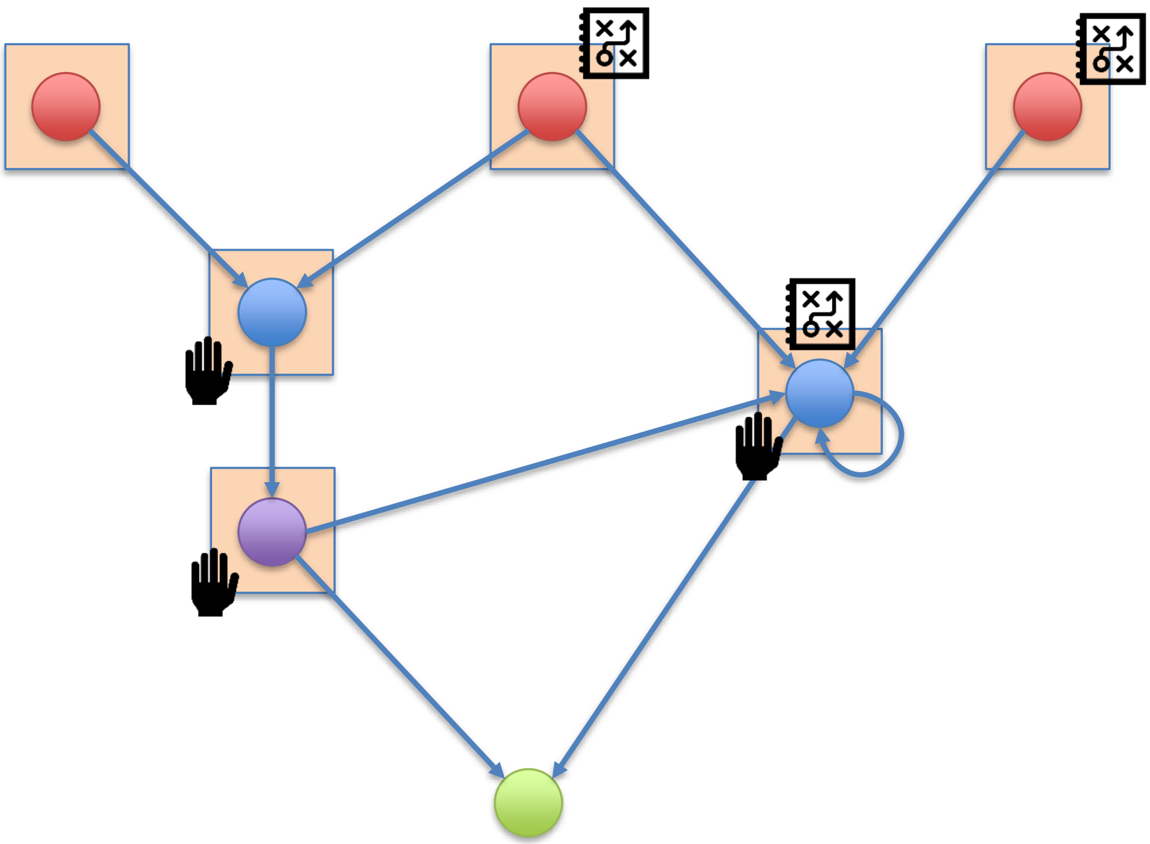
*an extension point: caching can be in-memory, distributed (e.g., Ehcache), …

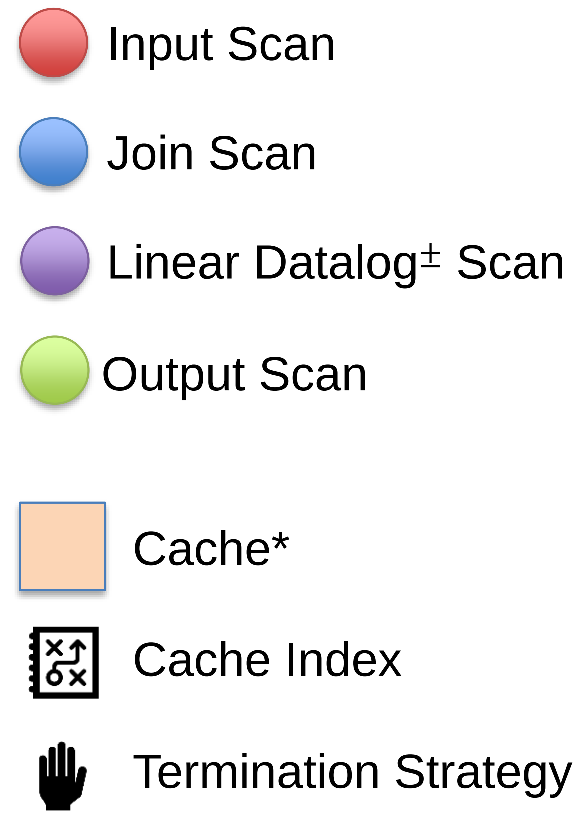
In-Memory Stream Processing



*an extension point: caching can be
in-memory, distributed (e.g., Ehcache), …

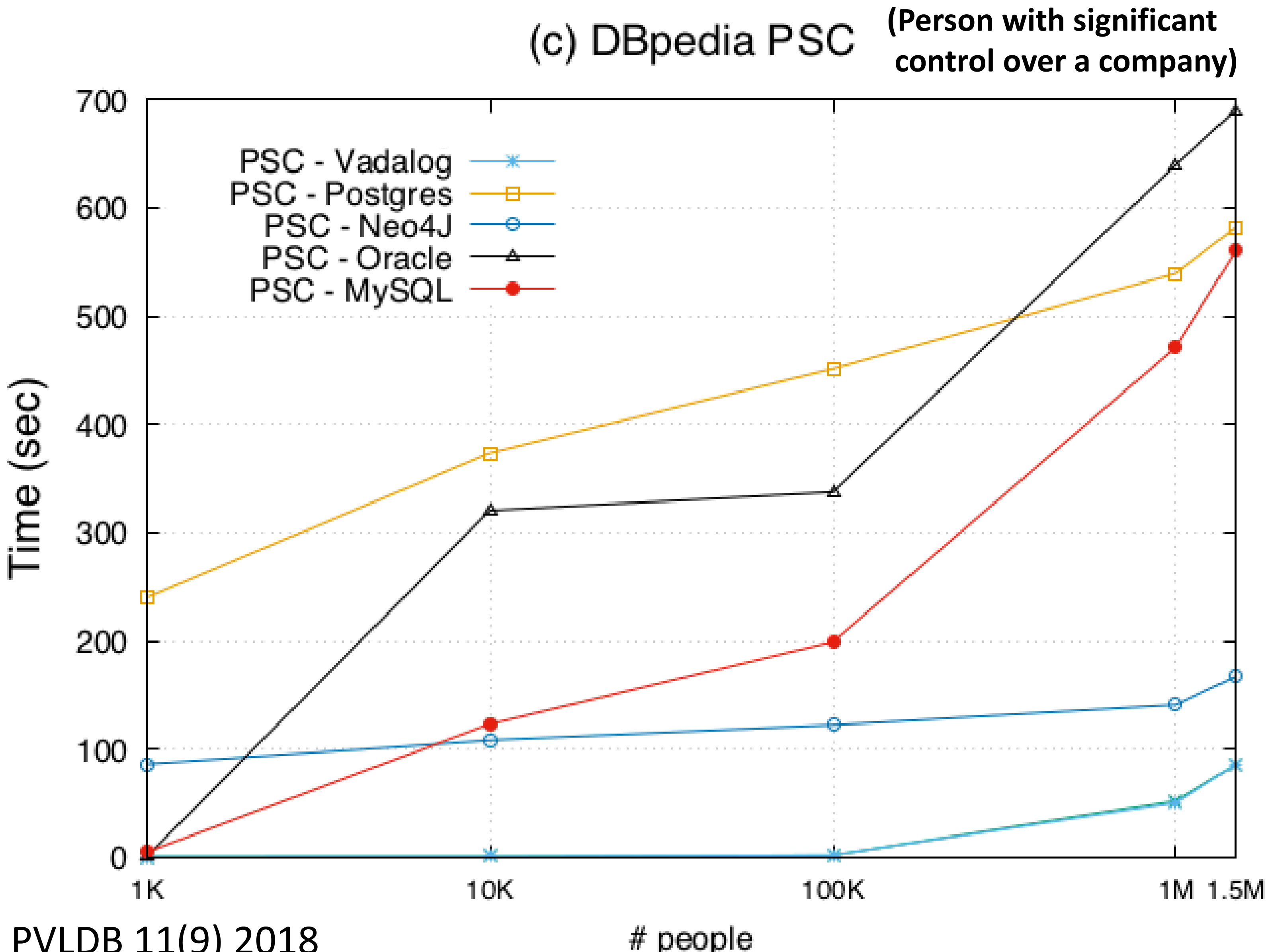
In-Memory Stream Processing



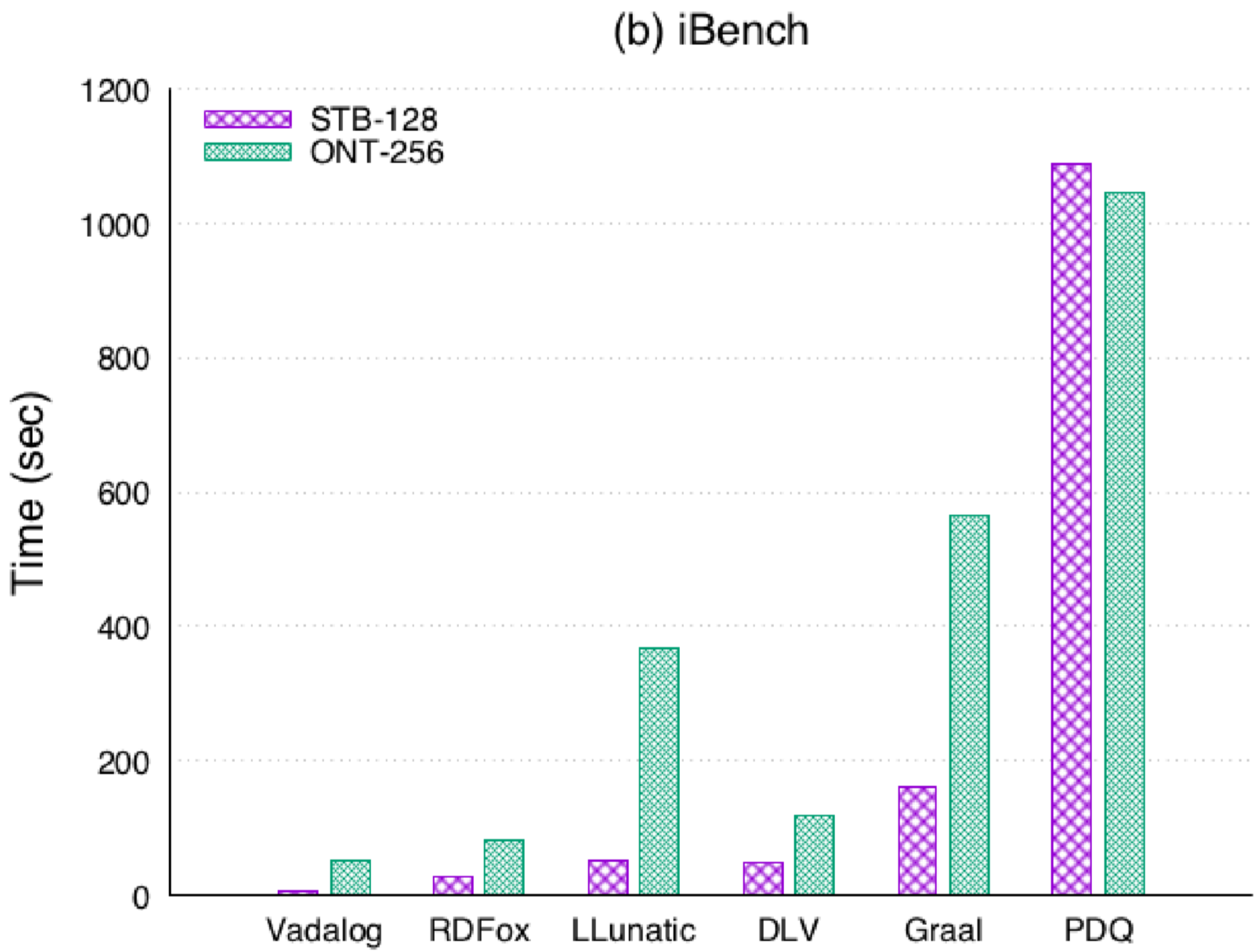


*an extension point: caching can be
in-memory, distributed (e.g., Ehcache), …

Performance



→ PVLDB 11(9) 2018



More benchmarks in Luigi Bellomarini, Emanuel Sallinger, Georg Gottlob: The Vadalog System: *Datalog-based Reasoning for Knowledge Graphs*. PVLDB 11(9) 2018

PAPER ON THE VADALOG LANGUAGE

- Marcelo Arenas, Georg Gottlob, Andreas Pieris: *Expressive languages for querying the semantic web*.
  ACM TODS 13:1-45, 2018.

PAPERS ON THE VADALOG SYSTEM

- Luigi Bellomarini, Georg Gottlob, Andreas Pieris, Emanuel Sallinger: *Swift Logic for Big Data and Knowledge Graphs*.
  International Joint Conference on Artificial Intelligence (IJCAI) 2017

- Luigi Bellomarini, Emanuel Sallinger, Georg Gottlob: The Vadalog System: *Datalog-based Reasoning for Knowledge Graphs*.
  PVLDB 11(9) 2018.

…

Some Applications

with two special partners/customers

Collaboration

1. Company Control	new approaches to classical problems – when does a company control another company?
2. Close Links	understanding whether companies are "too close" in terms of mutual stock participation for different purposes, e.g., for loan granting
3. Detection of Family Business	identifying families along with their ownerships, i.e., considering the family as the elementary control unit
4. Anonymization of Confidential Data	deciding whether a dataset respects complex confidentiality criteria (e.g., ISTAT) before publication and, if not, make it anonymous
5. Hybrid Data Science Pipelines	with different data sources, machine learning frameworks, programming languages, …

… more applications that we cannot talk about at this point

Banca d'Italia
Central Bank of Italy

Collaboration

1. Company Control	new approaches to classical problems – when does a company control another company?
2. Close Links	understanding whether companies are "too close" in terms of mutual stock participation for different purposes, e.g., for loan granting
3. Detection of Family Business	identifying families along with their ownerships, i.e., considering the family as the elementary control unit
4. Anonymization of Confidential Data	deciding whether a dataset respects complex confidentiality criteria (e.g., ISTAT) before publication and, if not, make it anonymous
5. Data Science: Hybrid pipelines	with different data sources, machine learning frameworks, programming languages, …

… more applications that we cannot talk about at this point

4. Anonymization of Confidential Data

We have a statistical survey about people that needs to be **anonymized**

Features	Dir	N	Rel	Rare	Vis	Tr	Sens
Fiscal code	✗						
Region					✗	✗	
Gender					✗	✗	
Age		✗	✗	✗	✗	✗	
Weight		✗		✗	✗		
Height		✗		✗	✗		
Marital status						✗	
Education						✗	
Work condition						✗	
Hospitalization							✗

Anonymization of direct features.

1: $\mathrm{Person}(f, a, r, e),\ l = 0 \rightarrow \exists i\ \mathrm{P}(i, a, r, e, l).$

Normalization of numeric features.

2: $\mathrm{P}(i, a, r, e, l),\ 0 < a \leq 20 \rightarrow \mathrm{P}(i, 10, r, e, l + 1).$

3: $\cdots\backslash$

4: $\mathrm{P}(i, a, r, e, l),\ 80 < a \leq 100 \rightarrow \mathrm{P}(i, 90, r, e, l + 1).$

Anonymization of related features ($a \times r$ and $a \times e$).

5: $\mathrm{P}(i, a, r, e, l),\ f = \mathrm{mcount}(i)/N \rightarrow \mathrm{FreqR}(a, r, f).$

6: $\mathrm{P}(i, a, r, e, l),\ \mathrm{FreqR}(a, r, f),\ f < F \rightarrow \exists x\ \mathrm{P}(i, a, x, e, l + 1).$

7: $\mathrm{P}(i, a, r, e, l),\ f = \mathrm{mcount}(i)/N \rightarrow \mathrm{FreqE}(a, e, f).$

8: $\mathrm{P}(i, a, r, e, l),\ \mathrm{FreqE}(a, e, f),\ f < F \rightarrow \exists x\ \mathrm{P}(i, a, r, x, l + 1).$

Anonymization of traceable features $a \times r \times e$.

9: $\mathrm{P}(i, a, r, e, l),\ f = \mathrm{mcount}(i)/N \rightarrow \mathrm{FreqLy}(a, r, e, f).$

10: $\mathrm{P}(i, a, r, e, l),\ \mathrm{FreqLy}(a, r, e, f),\ f < F \rightarrow \exists x \exists y\ \mathrm{P}(i, a, x, y, l + 1).$

Output.

11: $\mathrm{P}(i, a, r, e, l),\ i = \mathrm{argmax}(\langle i \rangle, l) \rightarrow \mathrm{Q}(a, r, e, l).$

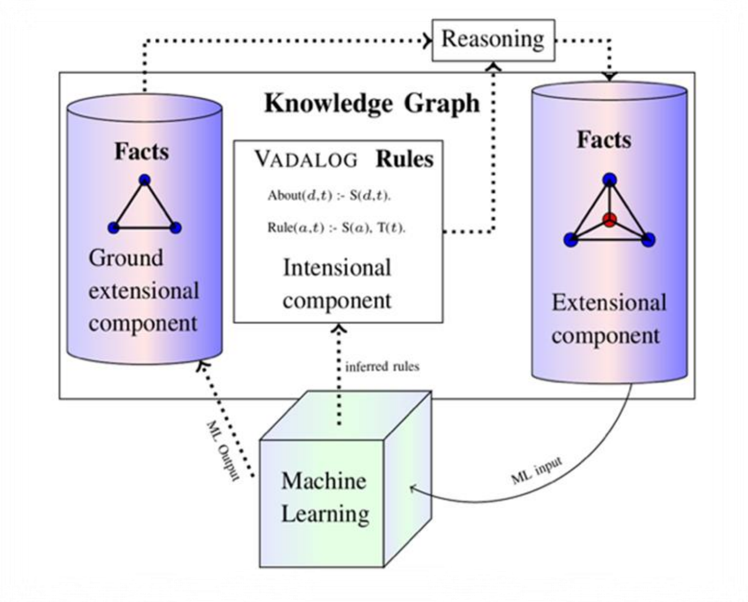
According to ISTAT guidelines, the survey features are classified in Direct, Numerical, Related, Rare, Visible, Traceable, Sensitive

On the variables:
Fiscal code (=SSN) (*f*), **age** (*a*), **region** (*r*), **education**(*e*)

5. Hybrid Data Science Pipelines

Building hybrid data science pipelines, including Vadalog, Python, Kafka, Flink, Dgraph, GraphX, ML, …

One approach for interaction between Jupyter, ML and **Vadalog**: Python native driver for Vadalog

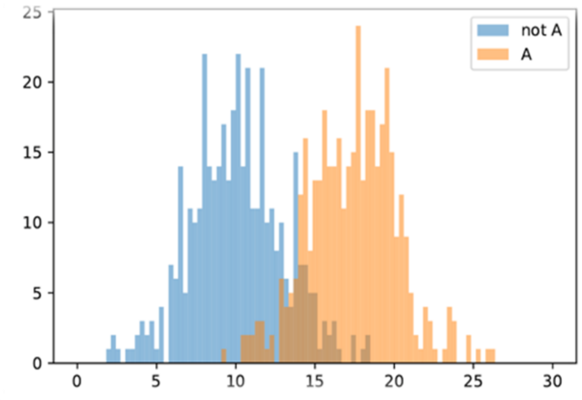


Example (next slide):
Company-type classification problem using “domain knowledge”: Italian SAE-Code

Calculate $*HighInc*, $*MaxInc*, $*MinInc* parameters in Python and pass them to the Vadalog **KG**

```
import pandas as pd
df = pd.read_csv("./dataset.csv", sep=\'t')
df_inc = df[df['INCOME'] > 0.0]
x = df_inc[df['NACE'] == 'A'].hist(bins=100)
y = df_inc[df['NACE'] != 'A'].hist(bins=100)
bins = np.linspace(0, 30, 100)
pyplot.hist(x, bins, alpha=0.5, label='not A')
pyplot.hist(y, bins, alpha=0.5, label='A')
pyplot.legent(loc='upper right')

# Parameters
high_income = inters_normal_hist(x, y)
max_income = df_inc.max()
min_income = df_inc.min()
```




```
from vadalog import vadalog
# Parameters binding
par = {'S': string, 'MinInc': min_income,
    'MaxInc': max_income,
    'HighInc': high_income}
vc = vadalog.connect()
resp = vc.evaluate('Nace_KG', par)
```

Nace KG (part):

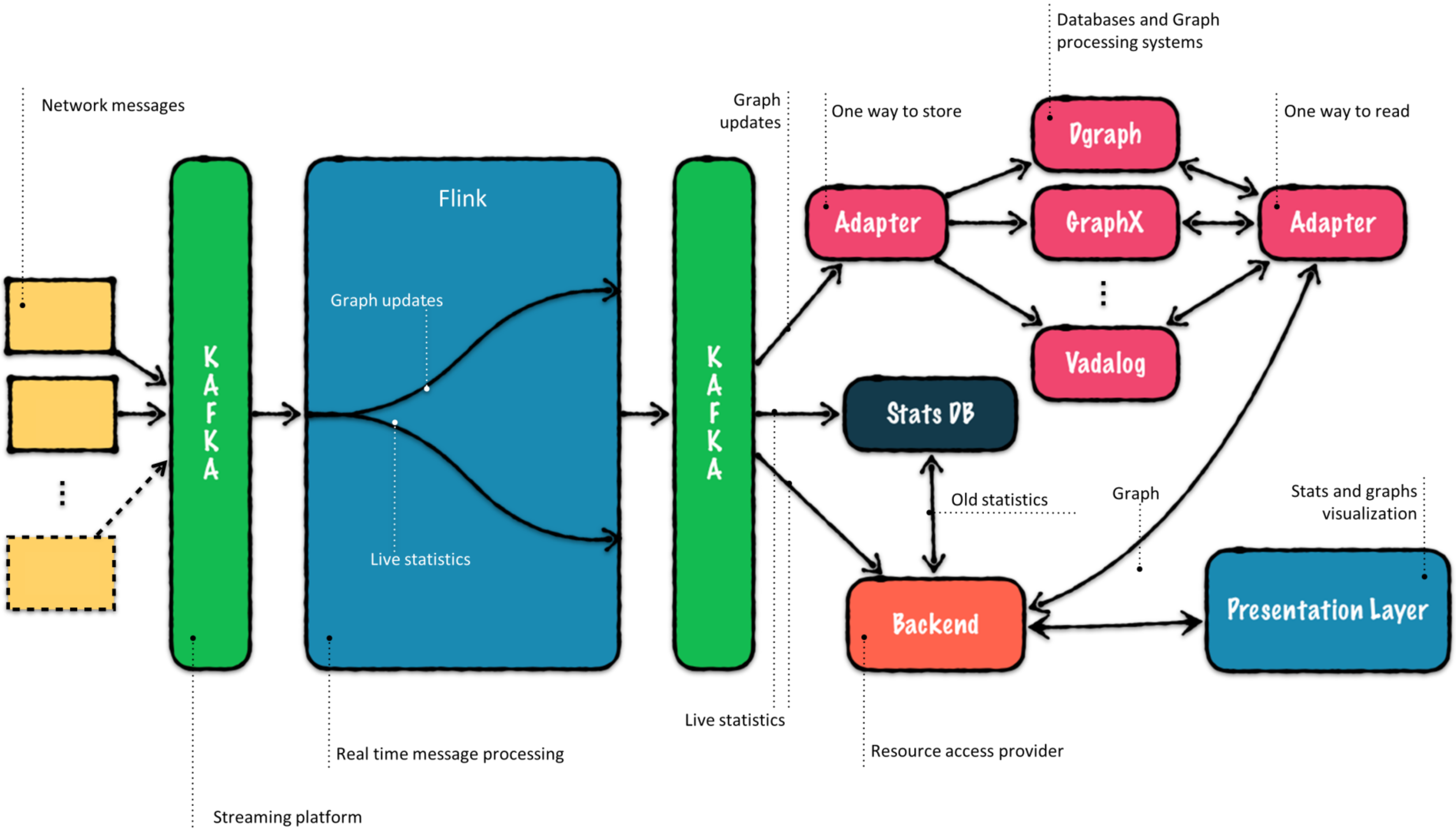
$$\mathrm{Income}(c,i),\ i > \$\mathrm{HighInc},\, p = (i - \$\mathrm{HighInc}) \times (0.5/(\$\mathrm{MaxInc} - \$\mathrm{HighInc})) + 0.5 \to \mathrm{IncomeProb}(c,p).$$

$$\mathrm{Income}(c,i),\ 0 < i \leq \$\mathrm{HighInc},\, p = (i - \$\mathrm{MinInc}) \times (0.5/(\$\mathrm{HighInc} - \$\mathrm{MinInc})) \to \mathrm{IncomeProb}(c,p).$$

$$\mathrm{Name}(c,n),\, j = \mathrm{contains}(n, \$\mathrm{S}) \to \mathrm{NameFound}(c,j).$$

$$\mathrm{IncomeProb}(c,p), \mathrm{NameFound}(c,j) \to \mathrm{inGroupA}(c, \max(p,j)).$$

5. Hybrid Data Science Pipelines



“ACME”

Collaboration

1. Entity Resolution

2. Similarity in Bipartite Graphs

3. Knowledge Graph Support

4. Computing Higher-Level Events and Signals on KG

5. Fact Enrichment and Verificiation on KGs

… more applications that we cannot talk about at this point

“ACME”

Collaboration

1. Entity Resolution

2. Similarity in Bipartite Graphs

3. Knowledge Graph Support

4. Computing Higher-Level Events and Signals on KG

5. Fact Enrichment and Verificiation on KGs

… more applications that we cannot talk about at this point

1. Entity Resolution

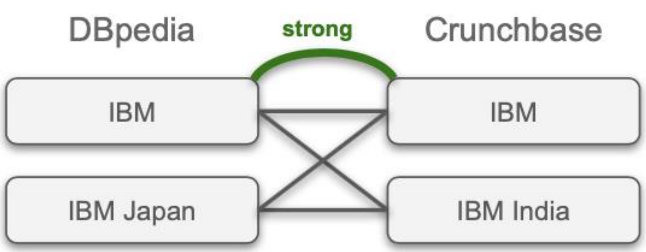
- Coverage of **Internal KG (IKG)Team**

  (via machine learning)

  and **Vadalog DeepReason Solution (DR)**

  - 7,994 company pairs linked by IKG
  - 16,379 company pairs linked by VADA
  - 10,413 company pairs identified by VADA as **strongly linked**

- Accuracy (1,200 manually inspected company pairs)



	Links		Strong Links	
	IKG	DR	IKG	DR
Precision	0.99	0.99	0.99	0.98
Recall	0.44	0.93	0.68	0.88
F1 Score	**0.61**	**0.97**	**0.80**	**0.92**

1. Entity Resolution

Entity resolution in three steps

1. Entity Blocking	• Reduce $O(n^2)$ complexity by comparing only entities with similar attributes • Extensive use of text cleaning functions to normalize input values
2. Entity Comparison	• Compare the similarity of every pair of entities for each of their attributes • Extensive us of text similarity functions • Normalize the similarity into a probability P_i, for each attribute i
3. Probability Computation	• For each pair of entities, compute the overall probability from the individual attribute probabilities P_i using the Naive Bayes formula $$P = \frac{\prod P_i}{\prod P_i + \prod(1 - P_i)}$$

1. Entity Blocking

compute a key for blocking by applying cleaning functions on attributes (e.g. names, addresses, urls, etc.)

```
key(Id, Key) :-
    entityName(Id, Name),
    Key = sim:trim(
          sim:removeNonWord(
          sim:removeDiacritics(
          sim:toLowerCase(Name)))).
```

establish the pairs of entities to be compared

```
block(DId, CId) :-
    key(DId, Key),
    key(CId, Key).
```

2. Entity Comparison

compute similarity per attribute (e.g. "name", "address", etc.)

```
attributeSimilarity(DId, CId, "name", Sim) :-
    block(DId, CId),
    entityName(DId, DName),
    entityName(CId, CName),
    Sim = sim:mongeElkan(DName, CName).
```

+ several special rules expressing specific knowledge,
e.g. about URL structure (zurich.ibm.com vs almaden.ibm.com)

3. Probability Computation

combine similarities (as probabilities) using Naive Bayes
with simple use of aggregates

```
overallProbability(DId, CId, OverallProbability) :-
    attributeSimilarity(DId, CId, Att, Prob),
    ProbProd = prod(Prob),
    InvProbProd = prod(1 - Prob),
    OverallProbability = ProbProd / (ProbProd + InvProbProd).
```

…or any other ML-based or statistics-based way to combine similarities


Thank You!