Knowledge Graphs and Enterprise AI
The Promise of an Enabling Technology

Georg Gottlob
Univ. of Oxford, TU Wien
& DeepReason.ai
Knowledge Graphs and Enterprise AI
The Promise of an Enabling Technology

Georg Gottlob
Univ. of Oxford, TU Wien
& DeepReason.ai
Knowledge Graphs as Large “World” KBs

Cyc [Lenat & Guha 1989]  
\[ \text{w}: \text{“comprehensive ontology and knowledge base of everyday common sense knowledge”} \].

Freebase [Bollacker et al. 2007]  
\[ \text{w}: \text{“online collection of structured data harvested from many sources, including user-submitted wiki contributions”} \].

Google Knowledge Graph [Singhal 2012] + K.Vault [Dong et al. 2014]  
\[ \text{w}: \text{“KB used by Google to enhance its search engine's search results with semantic-search information gathered from a wide variety of sources”} \].

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

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

- HSBC Advance
  - up to £10,000
- Saga Platinum
  - £8,500
- HSBC Credit Card
  - £12,000
- FINECO
  - up to EUR 10,000
- Bank Austria
  - up to EUR 20,000
- SANDBANK
  - £500
- M&S Bank
  - £8,000
- mbna
  - £12,500
- UniCredit Card
  - 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 **wrong data**.
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 **wrong data**.

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 most likely moved out.*

*(Such updates are often missing in the database)*

This rule can be used to update the database **before** 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”. Yikes, a fly agaric!
Desiderata for KGMS According to our Philosophy

No extra permanent data repository or database/DBMS
- Uses (possible multiple) existing 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 \( \text{argmin} \) (with grad. desc.), \( \text{eigenvector} \), \( \text{pagerank} \), \( \text{simrank} \), etc.
Knowledge Graph Management Systems

*a diverse new field – many systems with different capabilities*
<table>
<thead>
<tr>
<th><strong>Analysis along many dimensions possible</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AllegroGraph</strong></td>
</tr>
<tr>
<td><strong>GRAKN.AI</strong></td>
</tr>
<tr>
<td><strong>GP</strong></td>
</tr>
<tr>
<td><strong>VADALOG</strong></td>
</tr>
<tr>
<td><strong>neo4j</strong></td>
</tr>
<tr>
<td><strong>Stardog</strong></td>
</tr>
<tr>
<td><strong>metaphacts</strong></td>
</tr>
<tr>
<td><strong>MAANA</strong></td>
</tr>
<tr>
<td><strong>GNOSIS</strong></td>
</tr>
<tr>
<td>Company</td>
</tr>
<tr>
<td>-----------------</td>
</tr>
<tr>
<td>AllegroGraph</td>
</tr>
<tr>
<td>GRAKN.AI</td>
</tr>
<tr>
<td>Knowledge</td>
</tr>
<tr>
<td>VADALOG</td>
</tr>
<tr>
<td>neo4j</td>
</tr>
<tr>
<td>Stardog</td>
</tr>
<tr>
<td>metaphacts</td>
</tr>
<tr>
<td>MAANA</td>
</tr>
<tr>
<td>GNOSS</td>
</tr>
<tr>
<td>Principle Data Format / Backend</td>
</tr>
<tr>
<td>--------------------------------</td>
</tr>
<tr>
<td>AllegroGraph</td>
</tr>
<tr>
<td>Apache Cassandra</td>
</tr>
<tr>
<td>GRAKN.AI</td>
</tr>
<tr>
<td>VADALOG</td>
</tr>
<tr>
<td>neo4j</td>
</tr>
<tr>
<td>Stardog</td>
</tr>
<tr>
<td>metaphacts</td>
</tr>
<tr>
<td>MAANA</td>
</tr>
<tr>
<td>GNOSIS</td>
</tr>
<tr>
<td><strong>Analysis along many dimensions possible</strong></td>
</tr>
<tr>
<td>---------------------------------------------------</td>
</tr>
<tr>
<td><strong>AllegroGraph</strong></td>
</tr>
<tr>
<td><strong>GRAKN.AI</strong></td>
</tr>
<tr>
<td><strong>GP</strong></td>
</tr>
<tr>
<td><strong>VADALOG</strong></td>
</tr>
<tr>
<td><strong>neo4j</strong></td>
</tr>
<tr>
<td><strong>Stardog</strong></td>
</tr>
<tr>
<td><strong>metaphacts</strong></td>
</tr>
<tr>
<td><strong>MAANA™</strong></td>
</tr>
<tr>
<td><strong>GNOSS</strong></td>
</tr>
</tbody>
</table>

To our best knowledge, the most expressive reasoning language with tractability guarantees.
Vadalog KGMS Being Built at Oxford

• VADA = Value-Added DAta
• General architecture of VADALOG system
• Core reasoning language VADALOG = Warded Datalog + extensions
• Connectivity: Some plug-ins
special features:
argmin, sampling, graph libraries (e.g. separators), matrix ev, simrank, pagerank, ...
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 $\rightarrow$ 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

Source Schema $S$

Target Schema $T$

$\sum_{st}$

$\sum_t$

employee($Lastname$, $Firstname$, $Address$)

person($FirstName$, $Lastname$, $Birthdate$)

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

[Fagin, Kolaitis, Miller & Popa, 2003]; [Arenas et al., 2014]
Object Creation
e.g. in web data extraction

<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>
Object Creation
e.g. in web data extraction

<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>
Object Creation
e.g. in web data extraction

<table>
<thead>
<tr>
<th>T₁</th>
<th>T₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRODUCT</td>
<td>PRICE</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>
Object Creation
e.g. in web data extraction

table(T_1),
table(T_2),
sameColor(T_1,T_2),
isNeighbourRight(T_1,T_2) →
   \exists T \ textbox(T),
   \exists T \ contains(T,T_1),
   \exists T \ contains(T,T_2).

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

\[
\begin{align*}
\text{spouse1}(u, y1) & \land \text{spouse2}(u, y2) \land \text{start}(u, 1964) \land \text{end}(u, 1974) \\
\forall u, v, x, y. & \text{married}(u, v, x, y) \rightarrow \text{married}(v, u, x, y)
\end{align*}
\]

This symmetry rule for marriage intervals now becomes:

\[
\begin{align*}
\text{spouse1}(k1, taylor), & \land \text{spouse2}(k1, burton), \land \text{start}(k1, 1964), \land \text{end}(k1, 1974) \\
\text{spouse1}(k2, burton), & \land \text{spouse2}(k2, taylor), \land \text{start}(k2, 1964), \land \text{end}(k2, 1974)
\end{align*}
\]
### 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(_{\text{core}})</strong></td>
<td></td>
</tr>
<tr>
<td>professor (\sqsubseteq \exists \text{teachesTo})</td>
<td>(\forall X \ \text{professor}(X) \rightarrow \exists Y \ \text{teachesTo}(X,Y))</td>
</tr>
<tr>
<td>professor (\sqsubseteq \neg \text{student})</td>
<td>(\forall X \ \text{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>(\exists \text{hasTutor}) (\sqsubseteq \exists \text{teachesTo})</td>
<td>(\forall X \forall Y \ \text{hasTutor}(X,Y) \rightarrow \text{teachesTo}(Y,X))</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[∃]**: Full Datalog augmented with ∃-quantifier

Unfortunately:

**Theorem:** Reasoning \( (KB \models q) \) 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±**: Datalog[∃,⊥,¬strat, ...] subject to syntactic restrictions.

**Vadalog**: member of the Datalog± family admitting efficient reasoning methods.
Main Decidable Datalog\(^\pm\) Languages

- Weakly-(frontier-)guarded
- BTS
- Guarded
- Linear
- Datalog
- DL-Lite\(_R\)
- \(\mathcal{ELHI}\)
- Sticky
- Weakly-Sticky
- FUS
Datalog[$\exists,...$]

UNDECIDABLE

weakly frontier-guarded Datalog[$\exists,\bot,\neg$strat]

EXPTIME

Core Vadalog

= warded Datalog[$\exists,\bot,\neg$strat]

PTIME

Datalog[$\bot,\neg$strat]

Linear Datalog$^{\pm}[\exists,\bot]$

SPARQL + OWL2QL

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

EXPTIME

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

UNDECIDABLE

Data complexity

Core Vadalog

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

PTIME

NLOGSPACE

SPARQL + OWL 2QL

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

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

\[
T(X, Y, Z) \quad \rightarrow \quad \exists W \quad P(W, Z)
\]

\[
P(X, Y) \quad \rightarrow \quad \exists Z \quad Q(X, Z)
\]

Core Vadalog = warded Datalog[\(\exists, \bot, \neg\text{strat}\)]
Examples of Warded Datalog± Rules

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

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

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

<table>
<thead>
<tr>
<th>DL-Lite TBox</th>
<th>Representation in Vadalog</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DL-Lite\text{core}</strong></td>
<td></td>
</tr>
<tr>
<td>professor $\sqsubseteq \exists$ teachesTo</td>
<td>$\forall X \text{ professor}(X) \rightarrow \exists Y \text{ teachesTo}(X,Y)$</td>
</tr>
<tr>
<td>professor $\sqsubseteq \neg$ student</td>
<td>$\forall X \text{ professor}(X) \land \text{ student}(X) \rightarrow \bot$</td>
</tr>
<tr>
<td><strong>DL-Lite\text{R}(OWL 2 QL)</strong></td>
<td></td>
</tr>
<tr>
<td>hasTutor$^{-}$ $\sqsubseteq$ teachesTo</td>
<td>$\forall X \forall Y \text{ hasTutor}(X,Y) \rightarrow \text{ teachesTo}(Y,X)$</td>
</tr>
</tbody>
</table>
**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
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

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

**Probabilistic reasoning:** facts and rules can be adorned with
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
@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}).
Machine Learning, Big Data Analytics, NLP & Data Visualization

We are currently experimenting with different tools and different types of interfaces and interactions.
@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]:]"

Core Algorithms

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 management, query plan optimization
- Recent evaluation shows the system is extremely competitive
Interfaces (REST, JDBC, API, GUI, ...)

Parser

Logical optimizer

Planner

Execution plan optimizer

Query manager

Warded Datalog^±

Nearly-linear Datalog^±

Expressions evaluator

Aggregator

Probabilistic reasoning

Cache manager

Termination manager

In-memory indexer

Record manager

→ 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

(Person with significant control over a company)
PAPER ON THE VADALOG LANGUAGE

• Marcelo Arenas, Georg Gottlob, Andreas Pieris: *Expressive languages for querying the semantic web.*

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

... 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**

Anonymization of direct features.
1: \( \text{Person}(f, a, r, e), \ l = 0 \to \exists i \ P(i, a, r, e, l) \).

Normalization of numeric features.
2: \( P(i, a, r, e, l), \ 0 < a \leq 20 \to P(i, 10, r, e, l + 1) \).
3: \( \cdots \)
4: \( P(i, a, r, e, l), \ 80 < a \leq 100 \to P(i, 90, r, e, l + 1) \).

Anonymization of related features \((a \times r \text{ and } a \times e)\).
5: \( P(i, a, r, e, l), \ f = \text{mcount}(i) / N \to \text{FreqR}(a, r, f) \).
6: \( P(i, a, r, e, l), \ \text{FreqR}(a, r, f), \ f < F \to \exists x \ P(i, a, x, e, l + 1) \).
7: \( P(i, a, r, e, l), \ f = \text{mcount}(i) / N \to \text{FreqE}(a, e, f) \).
8: \( P(i, a, r, e, l), \ \text{FreqE}(a, e, f), \ f < F \to \exists x \ P(i, a, r, x, l + 1) \).

Anonymization of traceable features \(a \times r \times e\).
9: \( P(i, a, r, e, l), \ f = \text{mcount}(i) / N \to \text{FreqLy}(a, r, e, f) \).
10: \( P(i, a, r, e, l), \ \text{FreqLy}(a, r, e, f), \ f < F \to \exists x \exists y \ P(i, a, x, y, l + 1) \).

Output.
11: \( P(i, a, r, e, l), \ i = \text{argmax}(\langle i \rangle, l) \to Q(a, r, e, l) \).

<table>
<thead>
<tr>
<th>Features</th>
<th>Dir</th>
<th>N</th>
<th>Rel</th>
<th>Rare</th>
<th>Vis</th>
<th>Tr</th>
<th>Sens</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fiscal code</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Region</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Gender</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Age</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Weight</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Height</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Marital status</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Education</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Work condition</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hospitalization</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

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

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

```python
import pandas as pd
import matplotlib.pyplot as plt

# Read dataset
df = pd.read_csv("./dataset.csv", sep=\'\t\')

# Filter dataset
df_inc = df[df['INCOME'] > 0.0]

data = df_inc

# Calculate histogram
x = df_inc[df['NACE'] == 'A'].hist(bins=100)
y = df_inc[df['NACE'] != 'A'].hist(bins=100)
bins = np.linspace(0, 30, 100)

# Plot histograms
plt.hist(x, bins, alpha=0.5, label='not A')
plt.hist(y, bins, alpha=0.5, label='A')
plt.legend(loc='upper right')

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

# Import Vadalog
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):

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

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

Name(c, n), j = contains(n, SS) \rightarrow NameFound(c, j).

IncomeProb(c, p), NameFound(c, j) \rightarrow inGroupA(c, max(p, j)).
```
5. Hybrid Data Science Pipelines
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 Verification on KGs

... more applications that we cannot talk about at this point
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Entity Resolution</td>
</tr>
<tr>
<td>2.</td>
<td>Similarity in Bipartite Graphs</td>
</tr>
<tr>
<td>3.</td>
<td>Knowledge Graph Support</td>
</tr>
<tr>
<td>4.</td>
<td>Computing Higher-Level Events and Signals on KG</td>
</tr>
<tr>
<td>5.</td>
<td>Fact Enrichment and Verification on KGs</td>
</tr>
</tbody>
</table>

... 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)

<table>
<thead>
<tr>
<th></th>
<th>Links</th>
<th>Strong Links</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IKG</td>
<td>DR</td>
</tr>
<tr>
<td>Precision</td>
<td>0.99</td>
<td>0.99</td>
</tr>
<tr>
<td>Recall</td>
<td>0.44</td>
<td>0.93</td>
</tr>
<tr>
<td>F1 Score</td>
<td>0.61</td>
<td>0.97</td>
</tr>
</tbody>
</table>
# 1. Entity Resolution

## Entity resolution in three steps

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
</table>
| 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 use 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!