Reasoning on the Web: Scale and Uncertainty

Web Data Processing Systems

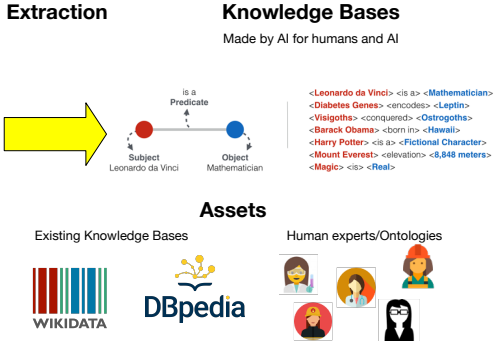
Jacopo Urbani

Department Computer Science
Vrije Universiteit Amsterdam, The Netherlands

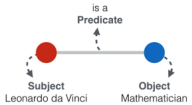
November 28, 2022

Recap

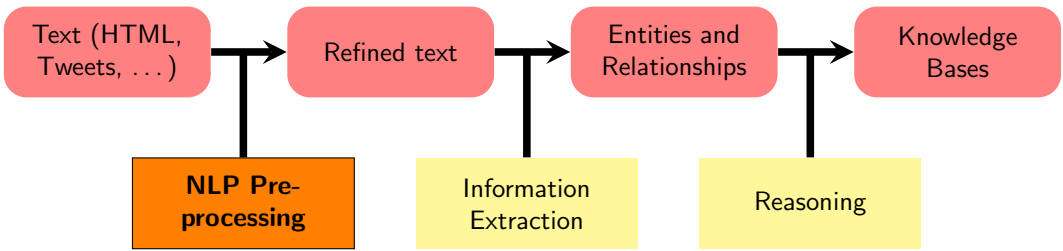
How can we use AI to **extract knowledge** from Web data?



Recap



<Leonardo da Vinci> <is a> <Mathematician>
<Diabetes Genes> <encodes> <Leptin>
<Visigoths> <conquered> <Ostrogoths>
<Barack Obama> <born in> <Hawaii>
<Harry Potter> <is a> <Fictional Character>
<Mount Everest> <elevation> <8,848 meters>
<Magic> <is> <Real>

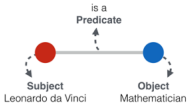


- Tokenization
- Lemmatization / Stemming

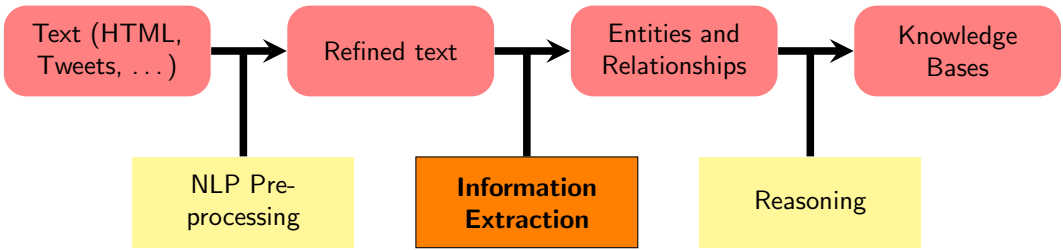
- POS
- Stop words removal

- Parsing
- . . .

Recap



<Leonardo da Vinci> <is a> <Mathematician>
<Diabetes Genes> <encodes> <Leptin>
<Visigoths> <conquered> <Ostrogoths>
<Barack Obama> <born in> <Hawaii>
<Harry Potter> <is a> <Fictional Character>
<Mount Everest> <elevation> <8,848 meters>
<Magic> <is> <Real>

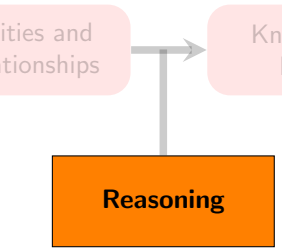


- Named Entity Recognition (NER)
- Named Entity Linking
- Relation Extraction
- Word Sense Disambiguation

Today

Outline

1. **Introduction to rule-based reasoning & ontologies**
2. Datalog
3. Beyond Datalog: Chase
4. Reasoning at a Web scale
5. Dealing with uncertainty



Logic-based Reasoning

Goal

With reasoning, our goal is to infer new knowledge applying a number of *logical steps*

Why using logic (and not deep learning)?

- Reasoning with logic is fully explainable
- KBs are symbolic artifacts and reasoning is a process that manipulates symbols
- Reasoning works at a large scale
- It is easy to input knowledge and observe the effect of it
- We don't know (yet) DL techniques that can do long sequences of reasoning steps

Important

One of the most exciting frontiers in AI consists of combining symbolic reasoning with deep learning! At the end of the lecture, we will discuss an example of such integration

Basic notions (1)

Symbols

We consider three types of symbols: *Variables* $\mathcal{V}$: $X, Y, Z, \ldots$; *Constants* $\mathcal{C}$: $a, b, c, \ldots$; *Predicates* $\mathcal{P}$: $p, q, r, \ldots$

Atoms

An *atom* is an expression of the form $p(t_1, \ldots, t_n)$ where $p \in \mathcal{P}$ and $t_1, \ldots, t_n \in \mathcal{C} \cup \mathcal{V}$. If an atom does not contain any variable, then it is a *fact*

Rules

A *rule* is an expression of the form $B_1, B_n \rightarrow H$ where $B_1, \ldots, B_n, H$ are atoms. The set $B_1, \ldots, B_n$ is called the *body* of the rule while H is the *head*. Let r be a rule. We denote its head as $\text{head}(r)$ and its body as $\text{body}(r)$

Mappings

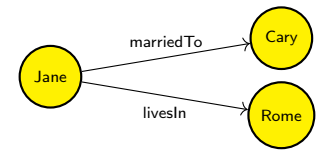
A *mapping* σ maps variables to constants. We apply σ as a postfix operator to atoms, rules, or sets thereof. For instance if $p(X, Y)$ and $\sigma = \{X \mapsto a, Y \mapsto b\}$, then $p(X, Y)\sigma = p(a, b)$

Basic notions (2)

From KBs to facts

Encoding a KB into a set of logical facts is easy

"Jane was living in Rome when her husband, Cary, . . ."



$\Rightarrow$ $livesIn(Jane, Rome)$
$marriedTo(Jane, Cary)$

Definition (database)

A *database* is a (finite) collection of facts

Rules: Where do they come from?

Typically, provided either by (human) users or by other processes. For instance,

Ontologies (e.g., [26])

$isA(X, Y) \wedge subclassOf(Y, Z) \rightarrow isA(X, Z)$

Data mining (e.g., [11])

$livesIn(X, Y) \wedge married(X, Z) \rightarrow livesIn(Z, Y)$

Data integration (e.g., [17])

$email(X, Y) \wedge email(Z, Y) \rightarrow X \approx Z$

Rules: Where do they come from?

...

Recursive querying (e.g., [21])

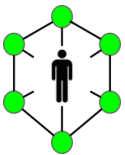
$$path(X, Y) \wedge path(Y, Z) \rightarrow path(X, Z)$$

Checking constraints (e.g., [23])

$$partOf(X, Y) \wedge partOf(Y, Z) \wedge partOf(Z, X) \rightarrow violation(X, Y, Z)$$

Knowledge completion (e.g., [7])

$$person(X) \rightarrow \exists Y.motherOf(Y, X)$$



Ontologies

Definition

An *ontology* is a formal explicit specification of a shared body of concepts

- Ontologies were initially studied (and still are) in philosophy. They constitute the study of nature of being, becoming, existence
- In computer science, they refer to something more concrete, namely to a formal description over a domain of discourse
- Ontologies are heavily used in AI to **represent knowledge**

Ontologies (example)

```
class-def animal                    % a
class-def plant                     % p
    subclass-of NOT animal          % t
class-def tree
    subclass-of plant               % t
class-def branch
    slot-constraint is-part-of      % b
        has-value tree
        max-cardinality 1
class-def defined carnivore         % carnivores are animals
    subclass-of animal
    slot-constraint eats            % that eat any other animals
        value-type animal
class-def defined herbivore         % herbivores are animals
    subclass-of animal, NOT carnivore % that are not carnivores, and
    slot-constraint eats            % they eat plants or parts of plants
        value-type plant OR (slot-constraint is-part-of has-value plant)
```

Option 1

$$tree(X) \rightarrow plant(X)$$

Option 2

$$isA(X, tree) \wedge subClassOf(tree, plant) \rightarrow isA(X, plant)$$

Ontological languages

Definition

Ontological languages are formal languages designed for defining ontologies in such a way that they can be interpreted without ambiguities

History

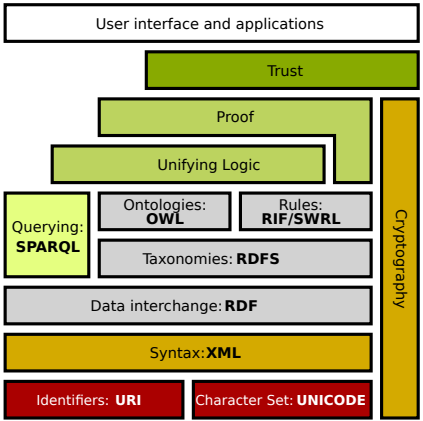
In the AI field of knowledge representation, ontological languages have a rich history (e.g., see `https://en.wikipedia.org/wiki/Ontology_language`)

Ontologies and the Web

Recently, the Web has emerged as a good use case for a massive and distributed application of ontologies

The Semantic Web

The ultimate goal of *Semantic Web* is to make the machine to understand the Internet data [5]



RDF Schema

RDF

RDF (which stands for "Resource Description Framework") is a data model what was standardized by W3C in 1999

Example

A RDF triple is a *subject,predicate,object* sequence of IRIs, Literals, or blank nodes

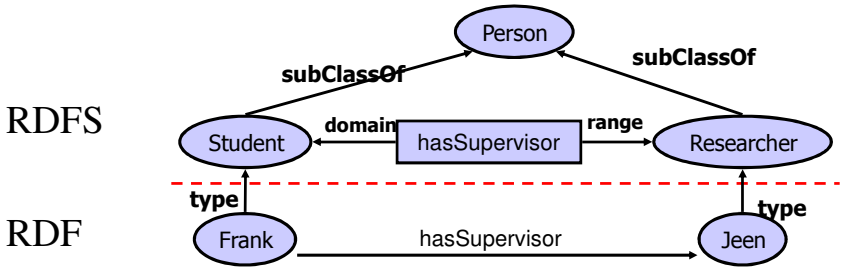
subject	`https://en.wikipedia.org/wiki/Dante_Alighieri`
predicate	`https://www.wikidata.org/wiki/Property:P19` (place of birth)
object	`https://en.wikipedia.org/wiki/Florence`

RDF Schema

RDF schema is a vocabulary for RDF, which was standardized by W3C in 2004

It is a vocabulary that can be used to model a domain with relations like *subClassOf*, *domain*, *range*, etc.

RDF Schema (example)



OWL

Limitations of RDFS

RDF Schema is not very expressive. For instance, we cannot state that `https://nl.wikipedia.org/wiki/Vrije_Universiteit_Amsterdam` and `https://www.vu.nl` refer to the same entity

Web Ontology Language

OWL (Web Ontology Language) is an ontological language designed to write ontologies to be published on the Web. It was standardized by W3C in 2004

OWL is more expressive than RDFS

- ⟨`isConnected rdf:type owl:TransitivePropery`⟩
- ⟨`daughterOf owl:inverseOf motherOf`⟩
- ⟨`Vrije_Universiteit owl:sameAs VU`⟩
- ...

OWL 2

In 2009, W3C standardized a new version of the language, which extends further the expressivity of the language

- Min/max cardinalities
- Reflexive properties
- ...

OWL profiles

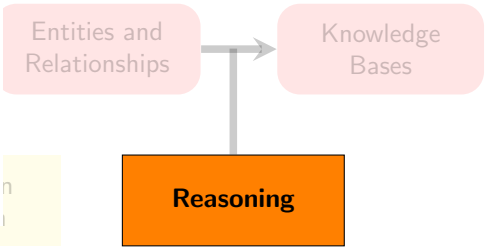
OWL 2 introduces *three* profiles:

- *OWL 2 EL:* suitable for applications employing ontologies that define very large numbers of classes and/or properties
- *OWL 2 QL:* designed to allow efficient query rewritings
- *OWL 2 RL:* when reasoning can be expressed with rules

Today

Outline

1. Introduction to rule-based reasoning & ontologies
2. **Datalog**
3. Beyond Datalog: Chase
4. Reasoning at a Web scale
5. Dealing with uncertainty



Datalog

Datalog [1] is a popular declarative query language developed in the '80s

- Queries are defined with sets of rules (expressive as it supports recursion)
- Query answering $\rightarrow$ execute the rules to infer all possible answers

Example

$lives(X, Y) \wedge marriedTo(X, Z) \rightarrow lives(Z, Y)$
$query(X) \rightarrow lives(mark, X)$

Safety

In Datalog, all rules are *safe*, i.e., every variable in the head must appear in the body.

Query answering with rules

Definition (BCQ)

A *boolean conjunctive query* (BCQ) is an expression of the form $\exists \mathbf{x}.p_1(\mathbf{t_1}) \wedge \ldots \wedge p_n(\mathbf{t_n})$ where $p_1(\mathbf{t_1}), \ldots, p_n(\mathbf{t_n})$ are atoms that contain only existentially quantified variables, i.e., appear in $\mathbf{x}$

Definition (Satisfiability)

Let K be a knowledge base of the form $\langle I, \Pi \rangle$, where I is a set of facts and Π is a set of rules. We say that K *satisfies* α, written $K \models \alpha$, if:

- $\alpha = p(\mathbf{t})$ and $p(\mathbf{t}) \in I$, or
- $\alpha = \alpha_1 \wedge \ldots \wedge \alpha_n$ and $K \models \alpha_1, \ldots, K \models \alpha_n$, or
- there is a rule $r \in \Pi$ such that $\alpha = \text{head}(r)\sigma$ and $K \models \text{body}(r)\sigma$

Definition (Query answering)

Let K defined as above and q be a boolean conjunctive query. Does $K \models q$?

Materialization

The most popular way to support query answering with rules is *materialization*

Input

Let I be the input set of facts, r be a rule be an expression of the form $B_1 \wedge B_2 \wedge \ldots B_n \rightarrow H$, and Π be the input set of rules

Mappings

Let σ be a mapping that replaces variables to constants, e.g., $\sigma = \{X \mapsto a, Y \mapsto b\}$. We use σ as suffix operator to replace variables, e.g., $married(X, Y)\sigma = married(a, b)$

Goal

$$\text{Let } r(I) = \{H\sigma \mid B_1\sigma, \ldots, B_n\sigma \in I\} \text{ and } \Pi(I) = \bigcup_{r \in \Pi} r(I)$$

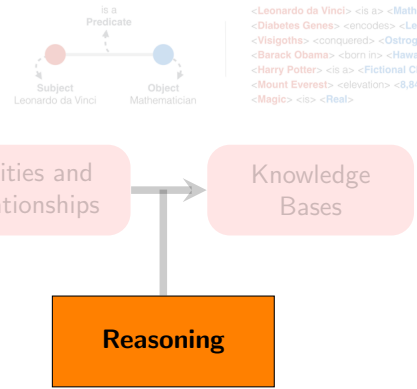
$$\text{Let } \Pi^0(I) = I \text{ and } \Pi^{i+1} = \Pi^i(I) \cup \Pi(\Pi^i(I)) \text{ for all } i > 0$$

Materialization: Compute $\Pi^\infty(I)$

Today

Outline

1. Introduction to rule-based reasoning & ontologies
2. Datalog
3. **Beyond Datalog: Chase**
4. Reasoning at a Web scale
5. Dealing with uncertainty



Existentially quantified rules

In some cases, it is useful to reason when something is unknown

In the database community, rules with existentially quantified variables in the head are called *tuple generating dependencies (TGDs)* [4]

Example

$$bicycle(X) \rightarrow \exists Y.hasPart(X, Y) \wedge wheel(Y)$$

In this case, even if we do not see the wheel, we still know it must exists

Problems

Unfortunately, reasoning with existentially quantified variables may be harder. In the worst-case, query answering becomes undecidable due to value invention

Fortunately, we know several conditions which are sufficient to guarantee decidability. These hold in many realistic scenarios [12]

Equality rules

Rules can also be used to state equivalence [4]. Such rules are called *equality generating dependencies (EGDs)*

Example

$$email(X, Y) \wedge email(Z, Y) \rightarrow X \approx Z$$

Techniques to support EGDs

- **Axiomatization.** Treat equality as a transitive, symmetric, and reflexive relation
- **Singularization.** Avoid explosion of derivations by rewriting joins
- **Replacement.** Use only one ID to refer to all synonyms

Chase

The *chase* [4] is a class of reasoning algorithms to materialize set of rules that may contain existentially quantified variables in the head

They all apply the rules bottom-up until saturation (i.e., first compute $\Pi^1(I)$, then $\Pi^2(I), \ldots$, until $\Pi^\infty(I)$)

Chase variants specify different conditions for satisfying the rules

- **Skolem chase**: "skolemize" existentially quantified variables and then treat them like all other variables
- **Restricted chase**: Introduce new values only if there are not partial instantiations of the head

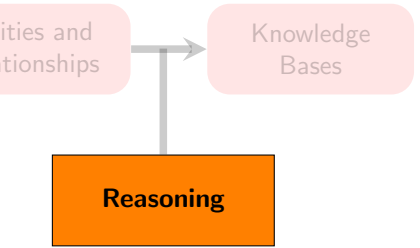
What's different?

All chase variables compute models which are suitable for query answering. However, there are cases when the restricted chase terminates while the skolem chase does not

Today

Outline

1. Introduction to rule-based reasoning & ontologies
2. Datalog
3. Beyond Datalog: Chase
4. **Reasoning at a Web scale**
5. Dealing with uncertainty



Scalability of reasoning

Despite the tractability of data complexity, reasoning over very large KBs can be time consuming (or require much computational resources)

Problems

Given I (the database) and Π (the ruleset), we view $\Pi^{\infty}(I)$ as $\Delta_0 \cup \Delta_1 \cup \Delta_2 \cup \ldots \cup \Delta_n$ where $\Delta_0 = I$ and $\Delta_i, i > 0$ contains all the facts obtained by applying a rule in Π on $\cup_{j<i}\Delta_j$

1. I can be too large to be stored on one machine
2. Π can be too large to be stored on one machine
3. Computing Δ_i can be time consuming
4. n can be very large
5. The problem is PTIME-complete (hence computation cannot be always parallelized)

Scalability of reasoning

Despite the tractability of data complexity, reasoning over very large KBs can be time consuming (or require much computational resources)

What can we do?

- Distribute (parallelize) the computation
  1. Rule parallelism (rules are mapped to different processors) e.g., RDFOx [20]
  2. Data parallelism (data partitions are mapped to different processors), e.g., WebPIE [26]
- **Compression (VLog [7])**

Techniques for Scalable Materialization

Parallelization and/or distribution is the mainstream approach to improve scalability

Rule parallelism
(e.g., RDFOx)

where $r_1, r_2, r_3, r_4 \in \Pi$

Data parallelism
(e.g., WebPIE)

where $r \in \Pi$ and $I_1 \cup I_2 \cup I_3 \cup I_4 = I$

Techniques for Scalable Materialization

Parallelization and/or distribution is the mainstream approach to improve scalability

Advantages

- Data parallelism addresses Problem (1) (I can be too large)
- Data parallelism can also address Problem (3) (Δ_i can be time-consuming)
- Both rule and data parallelism address Problem (2) (Π^∞ can be too large)
- Rule parallelism can lead to good load balancing

Disadvantages

- Parallelize and/or distribute a computation has a substantial engineering cost
- Clusters can be expensive
- It may not be possible to share data (privacy)
- May be not efficient due to Problem (5) (PTIME-completeness)

Compressing Materialization

Compression is an alternative, possibly complementary, way to improve the scalability of rule-based reasoning.

Advantages

- We can reduce the input size, hence addressing Problems (1) and (2)
- Compression addresses also Problem (3)
- In contrast to parallelism, we do not suffer from Problem (5)
- It can be combined with parallelism to further improve the performance

VLog: A rule-based engine for KGs

Motivation

Compression can improve the scalability of rule-based reasoning

- Reduce input size, hence can store larger inputs
- If the input is smaller, computation is faster
- In contrast to parallelism, we do not suffer from the P-hardness
- It can be used in combination with parallelism

Key features of VLog

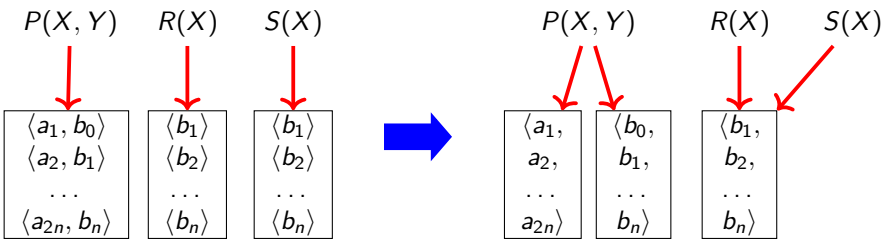
- Supports materialization with Datalog, skolem and restricted chase
- Supports negation via stratification
- Supports equality rules via axiomatization, singularization, and replacement
- Publicly available with open-source license https://github.com/karmaresearch/vlog

VLog: A rule-based engine for KGs

Main idea

The main idea behind VLog is to store the data using columns instead of rows

$$P(X, Y) \wedge R(X) \rightarrow S(X)$$



Problem

How to deal with updates? **Work in append-only mode**

Three main advantages

Advantage 1: Structure sharing

Instead of copying columns, we can store pointers (ok due append-only mode)

Advantage 2: Better compression

$\langle \underbrace{b, b, ..., b}_{n} \rangle \rightarrow \langle b \times n \rangle$ (from O(n) to O(1) storage)

$\langle b_1, b_2, \ldots, b_n \rangle \rightarrow \langle b_i \mid 1 \leq i \leq n \rangle$ same as above

Advantage 3: Avoid duplicate derivations

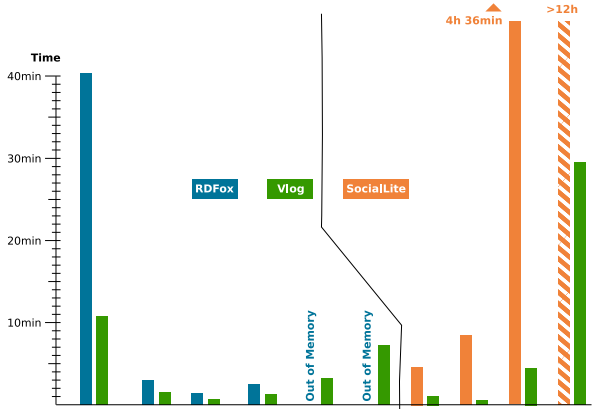
Consider rules $r_1 : P(X, Y) \rightarrow Q(Y, X)$ and $r_2 : Q(X, Y) \rightarrow P(Y, X)$.
If the database contains $P(c_1, c_2)$ and r_1 inferred $Q(p_{c_2}, p_{c_1})$, where p_{c_i} is a pointer to c_i, then skip r_2 on $Q(p_{c_2}, p_{c_1}) \Rrightarrow$ **avoided inference of** $|c_i|$ **duplicates**

VLog: Evaluation (1)

Competitors:

- RDFOx [20] (ontological reasoning)
- SociaLite [21] (graph analysis)

VLog outperforms the other systems, often significantly

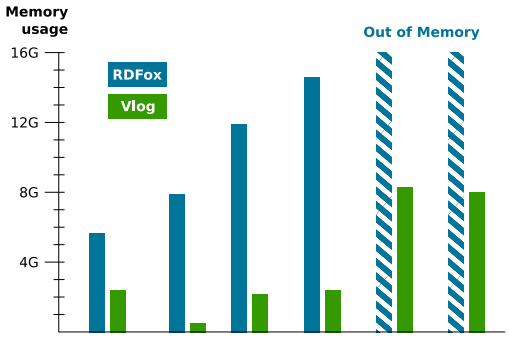


	Claros	DBpedia	LUBM-1K		LUBM-5K		Twitter	Google Webgraph		
Rules	2689	9396	170	202	170	202	2	5	5	5
Input tuples	19M	112M	133M		691M		1.75M	1M	2M	5M
Derived tuples	89M	33M	172M	197M	815M	994M	23M	31M	158M	769M

VLog: Evaluation (2)

Best case: VLog uses 14X less RAM

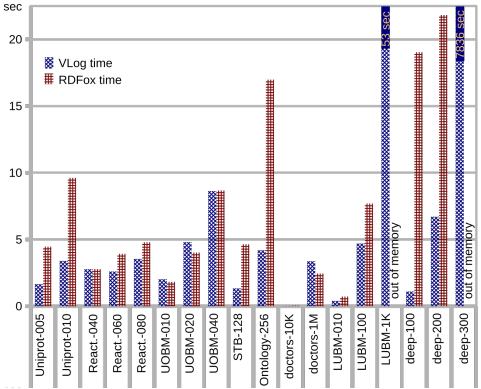
Worst case: VLog uses 2X less RAM

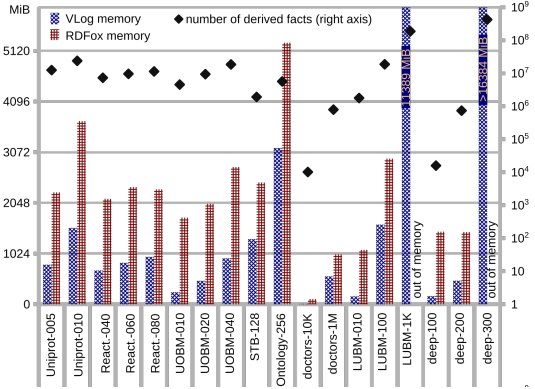


	Claros	DBpedia	LUBM-1K		LUBM-5K	
Rules	2689	9396	170	202	170	202
Input tuples	19M	112M	133M		691M	
Derived tuples	89M	33M	172M	197M	815M	994M

VLog: Evaluation (3)

VLog outperforms the competitors also with the restricted chase





GLog: Avoiding deriving duplicates

The derivation of duplicates is a problem that can significantly slow down the performance

Addressing this issue motivated the development of **GLog** [25]– a spinoff of VLog (https://www.github.com/karmaresearch/glog)

GLog proposes a new data structure, called *Trigger Graphs*, to perform materialization without generating (most) duplicates

GLog: Avoiding deriving duplicates

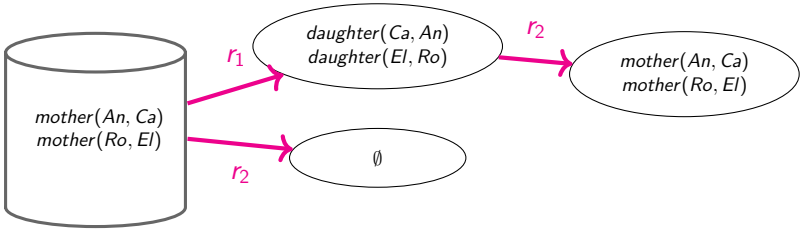
Example

Consider the rules

$$mother(X, Y) \rightarrow daughter(Y, X) \quad (r_1)$$
$$daughter(X, Y) \rightarrow mother(Y, X) \quad (r_2)$$

and $I = \{Mother(Anna, Carla), Mother(Rose, Elena)\}$

Trigger Graph



GLog: Performance Comparison

Inputs (#facts (millions),#rules)

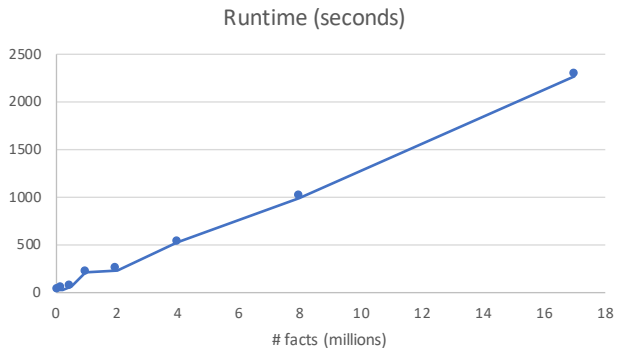
LUBM/L (var/170), LUBM/LE (var/561), UOBM (2/561), DBPedia (29/9.3k) Claros/LE (14/2.7k)

	Runtime (seconds)			*RAM (GB)*		
	GLog	*VLog*	*RDFOx*	*GLog*	*VLog*	*RDFOx*
LUBM/L	**1.0**	1.5	23	**0.26**	0.33	2.3
LUBM/LE	**16.1**	170	116	**1.3**	2.7	3.1
UOBM	**2.6**	7.3	10	**0.3**	1.0	0.78
DBPedia	**19**	41.6	64	1.3	**0.8**	3.2
Claros/LE	**1054**	2771	*TO*	48	**11.8**	NA

GLog: Scalability

Hardware and Inputs

PC with Intel Xeon E5 and 256GB RAM; LUBM KBs KB with up to **17 billion facts**

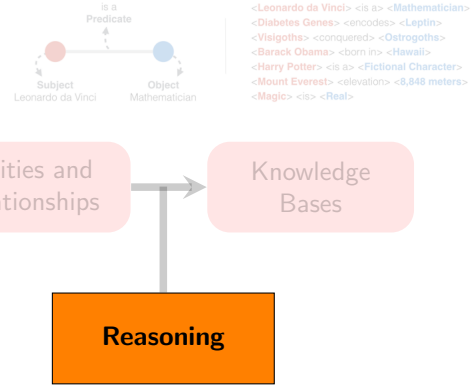


No other rule engine can scale up to such large KBs

Today

Outline

1. Introduction to rule-based reasoning & ontologies
2. Datalog
3. Beyond Datalog: Chase
4. Reasoning at a Web scale
5. **Dealing with uncertainty**



Probabilistic Reasoning

Current extraction techniques introduce **uncertainty**

Example

"Jane was living in Rome when her husband, Cary, . . . "

$\Downarrow$

Jane livesIn Rome (confidence 0.9)
Jane marriedTo Cary (confidence 0.6)

Probabilistic Reasoning

A well-known approach to blend uncertainty with reasoning is the **possible world semantics (PWS)**, which is used by many probabilistic databases

Possible World Semantics (1)

Main idea

Instead of having a database of certain *facts*, we have a database of facts and probabilities that the facts are true [22]

Example

Instead of having $I = \{livesIn(Jane, Rome), marriedTo(Jane, Cary)\}$, we set
$I = \{\langle livesIn(Jane, Rome), 0.9\rangle, \langle marriedTo(Jane, Cary), 0.6\rangle\}$

Possible Worlds

A database does not describe a single world, but multiple ones

$I_1 = \{livesIn(Jane, Rome), marriedTo(Jane, Cary)\}$ $\quad Prob(I_1) = 0.9 \times 0.6$
$I_2 = \{livesIn(Jane, Rome)\}$ $\quad Prob(I_2) = 0.9 \times 0.4$
$I_3 = \{marriedTo(Jane, Cary)\}$ $\quad Prob(I_3) = 0.1 \times 0.6$
$I_4 = \emptyset$ $\quad Prob(I_4) = 0.1 \times 0.4$

Notice that $Prob(I_1) + Prob(I_2) + Prob(I_3) + Prob(I_4) = 1$

Query Answering under Possible World Semantics

Query answering [22]

Since the database is a set of possible worlds, answering a query returns:

- all answers which are satisfied by a possible world
- for each answer, the probability that the answer is true in a randomly chosen world

Example

Recall the previous example. What is the probability that $\mathit{marriedTo}(\mathit{Jane}, \mathit{Cary})$ is true?

Answer: This fact is true in I_1 and I_3, hence the probability is $0.9 \times 0.6 + 0.1 \times 0.6$

Data complexity of query answering

Computing the output probabilities is #P-hard in the worst case, which is an intractable complexity class

Fortunately, there is a subset of queries, called safe queries, where the computation is P. These queries can be recognized by a series of syntactic checks

Towards Neuro-symbolic Probabilistic Reasoning: DeepProbLog

So far, the probabilities associated to the facts are part of the input. What if the probabilities should be *learned*, e.g., by a neural network?

DeepProbLog [18]

DeepProbLog is a neural-symbolic reasoner where queries are defined by rules and some predicates are mapped to neural networks

The neural networks produce probabilistic facts that are considered during reasoning. Those probabilities are no longer fixed, but "learned" by the neural networks

Symbolic reasoning is intertwined with the statistical predictions of neural networks, giving raise a paradigm called **neural-symbolic** reasoning

Towards Neuro-symbolic Probabilistic Reasoning: DeepProbLog

Challenges

1. *How can symbolic reasoning help the learning of the neural predicates?*

   In DeepProbLog, neural networks are trained with gradient descent, which is computed from the reasoning traces (**learning by entailment**)

Learning by entailment

Definition (lineage)

The *lineage* of a query answer is a propositional formula which says which fact in the database must be true for the answer to be satisfied

Example

Let $I = \{\langle burglary, p_b\rangle, \langle earthquake, p_e\rangle, \langle at\_home(mary), p_{am}\rangle, \langle at\_home(john), p_{aj}\rangle\}$, Π be a program with the rules

$$earthquake \rightarrow alarm \tag{1}$$

$$burglary \rightarrow alarm \tag{2}$$

$$alarm, at\_home(X) \rightarrow calls(X) \tag{3}$$

and the query q be $calls(mary)$. The lineage of q is $(burglary \vee earthquake) \wedge at\_home(mary)$

Learning by entailment

Computing the lineage is useful because it allows us to compute the probability of a query answer, e.g., using *weighted model counting*

If p_b, p_e, p_{am}, and p_{aj} are not fixed, then the lineage can be seen as a function that depends on parameters $p_b, \ldots$

DeepProbLog proposes the usage of *gradient semirings* to learn "good" values of p_b and other learnable parameters. A semiring is a mathematical structure which computes, for each fact, the corresponding probability and the gradient w.r.t. the parameters the probability depends on.

Example

Assume that $p_b = 0.1, p_e = 0.2$, and $p_{am} = 0.5$. The probability of *calls*(*mary*) is 0.14. From our training data, we know it should be 1. So we use the semiring to compute $\frac{\partial \mathcal{L}}{\partial p_e}$, $\frac{\partial \mathcal{L}}{\partial p_b}$, etc. and update the underlying neural networks

Towards Neuro-symbolic Probabilistic Reasoning: DeepProbLog

Challenges

1. *How can symbolic reasoning help the learning of the neural predicates?*

   In DeepProbLog, neural networks are trained with gradient descent, which is computed from the reasoning traces (**learning by entailment**)

2. *How can we implement query answering efficiently?*

   DeepProbLog does not address this problem directly, hence it has scalability issues, especially on the grounding of formulae [24]

   Scallop [16] addresses this issue with a top-k approximation

Scallop's main idea (informal)

The problem in more detail

Each answer for a given query is inferred by a sequence of rules applications on the input. Each sequence is part of the lineage of the answer

We can view the *lineage* of an answer of the disjunction of all its proofs. For instance, let a be an answer of a query. Then,

$$lineage(a) = proof_1 \vee proof_2 \vee proof_3 \ldots$$

The problem is that computing **all** proofs may be expensive. This hinders the computation of the marginal probability of the answers

Proposed solution

Scallop computes an approximated lineage by keeping only the k proofs with the highest probability. This improves the runtime of learning and inference, but it is an approximation without known error bounds

Trigger Graphs (TGs) for Probabilistic Reasoning

In a recent work (which will appear in SIGMOD 2023), we have shown how TGs (and GLog) are ideal for computing the lineage

In a nutshell

Instead of grounding like DeepProbLog and Scallop, we materialize the input storing all possible inferences in the TG. The resulting TG is a compact representation of the full probabilistic model

To obtain the lineage, we simply traverse the TG backwards and compute the groundings for each answer on-the-fly

Trigger Graphs (TGs) for Probabilistic Reasoning

Some empirical evidence (1)

The table below reports the total runtime of queries in the LUBM benchmark [13]

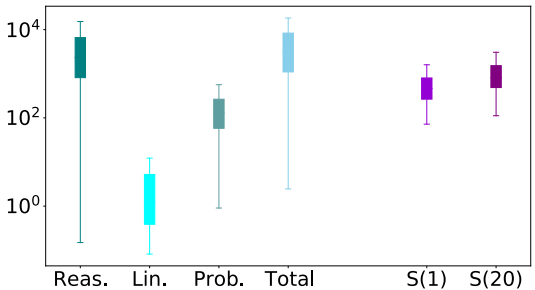
Query	GLog	vProbLog [24]	Scallop	DeepProbLog
Q1	**49ms**	587ms	1.3s	59ms
Q2	**383ms**	7.2s	NA	NA
Q3	**38ms**	306ms	729ms	NA
...			...	
Q12	**387ms**	10.6s	165s	78ms
Q13	**176ms**	541ms	30s	NA
Q14	**273ms**	337ms	326ms	150ms

Main conclusion: GLog outperforms (often significantly) prior art in terms of runtime

Trigger Graphs (TGs) for Probabilistic Reasoning

Some empirical evidence (2)

Runtime (ms) queries in VQAR (neuro-symbolic benchmark). Left is GLog (reasoning, computing lineage and probabilities, total). Right is Scallop with topk equal=1 and 20

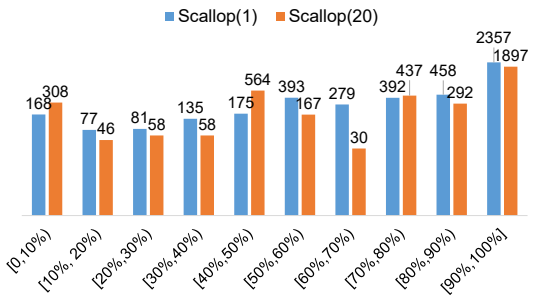


Conclusion: Despite GLog computes exact reasoning, runtimes are still competitive

Trigger Graphs (TGs) for Probabilistic Reasoning

Some empirical evidence (3)

Is approximating worthwhile? The graph below shows the fraction of the 5494 answers for which the approximated probability differs from the exact one (computed by GLog)



Conclusion: With Scallop, an inaccurate probability is returned for a high number of answers. If accuracy is paramount, then GLog is the only scalable solution

References I

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of databases*. Vol. 8. Addison-Wesley Reading, 1995.

[2] M. Banko, O. Etzioni, and T. Center. "The Tradeoffs Between Open and Traditional Relation Extraction.". In: *ACL*. Vol. 8. 2008, pp. 28–36. (Visited on 11/02/2016).

[3] M. Banko et al. "Open information extraction from the web". In: *IJCAI*. 2007, pp. 2670–2676.

[4] M. Benedikt et al. "Benchmarking the chase". In: *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*. 2017, pp. 37–52.

[5] T. Berners-Lee, J. Hendler, and O. Lassila. "The semantic web". In: *Scientific american* 284.5 (2001), pp. 34–43.

References II

[6] S. Brin. "Extracting patterns and relations from the world wide web". In: *International Workshop on The World Wide Web and Databases*. Springer, 1998, pp. 172–183. (Visited on 11/06/2016).

[7] D. Carral et al. "VLog: A Rule Engine for Knowledge Graphs". In: *The Semantic Web – ISWC 2019*. 2019, pp. 19–35.

[8] L. Cui, F. Wei, and M. Zhou. "Neural Open Information Extraction". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, 2018, pp. 407–413.

[9] F. Dernoncourt, J. Y. Lee, and P. Szolovits. "NeuroNER: an easy-to-use program for named-entity recognition based on neural networks". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. 2017, pp. 97–102.

References III

[10] J. R. Finkel, T. Grenager, and C. Manning. "Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling". In: *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. ACL '05. 2005, pp. 363–370.

[11] L. Galárraga et al. "Fast rule mining in ontological knowledge bases with AMIE+". In: *The VLDB Journal* 24.6 (2015), pp. 707–730.

[12] B. C. Grau et al. "Acyclicity Notions for Existential Rules and Their Application to Query Answering in Ontologies". In: *Journal of Artificial Intelligence Research* (2013), pp. 741–808.

[13] Y. Guo, Z. Pan, and J. Heflin. "LUBM: A benchmark for OWL knowledge base systems". In: *Web Semantics: Science, Services and Agents on the World Wide Web* 3.2 (2005), pp. 158–182.

References IV

[14] M. A. Hearst. "Automatic acquisition of hyponyms from large text corpora". In: *Proceedings of the 14th conference on Computational linguistics-Volume 2*. 1992, pp. 539–545. (Visited on 11/06/2016).

[15] J. Hoffart et al. "Robust disambiguation of named entities in text". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 2011, pp. 782–792.

[16] J. Huang et al. "Scallop: From Probabilistic Deductive Databases to Scalable Differentiable Reasoning". In: *Advances in Neural Information Processing Systems*. Vol. 34. 2021, pp. 25134–25145.

[17] B. Kruit, H. He, and J. Urbani. "Tab2Know: Building a Knowledge Base from Tables in Scientific Papers". In: *The Semantic Web – ISWC 2020*. 2020, pp. 349–365.

References V

[18] R. Manhaeve et al. "Neural probabilistic logic programming in DeepProbLog". In: *Artificial Intelligence* 298 (2021), p. 103504.

[19] A. K. McCallum. "MALLET: A Machine Learning for Language Toolkit". http://mallet.cs.umass.edu. 2002.

[20] Y. Nenov et al. "RDFox: A highly-scalable RDF store". In: *International Semantic Web Conference*. 2015, pp. 3–20.

[21] J. Seo, S. Guo, and M. Lam. "SociaLite: Datalog extensions for efficient social network analysis". In: *2013 IEEE 29th International Conference on Data Engineering (ICDE)*. 2013, pp. 278–289.

[22] D. Suciu et al. "Probabilistic Databases". In: *Synthesis Lectures on Data Management* 3.2 (2011), pp. 1–180.

References VI

[23] N. Tandon et al. "Commonsense in Parts: Mining Part-Whole Relations from the Web and Image Tags". In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*. 2016, pp. 243–250.

[24] E. Tsamoura, V. Gutiérrez-Basulto, and A. Kimmig. "Beyond the Grounding Bottleneck: Datalog Techniques for Inference in Probabilistic Logic Programs". In: *AAAI*. 2020, pp. 10284–10291.

[25] E. Tsamoura et al. "Materializing Knowledge Bases via Trigger Graphs". en. In: *Proceedings of the VLDB Endowment (PVLDB)* 14.6 (2021), p. 943.

[26] J. Urbani et al. "WebPIE: A Web-scale Parallel Inference Engine using MapReduce". In: *Web Semantics: Science, Services and Agents on the World Wide Web* 10 (2012), pp. 59–75.

References VII

[27] V. Yadav and S. Bethard. "A survey on recent advances in named entity recognition from deep learning models: 27th International Conference on Computational Linguistics, COLING 2018". In: *COLING 2018 - 27th International Conference on Computational Linguistics, Proceedings*. COLING 2018 - 27th International Conference on Computational Linguistics, Proceedings (2018), pp. 2145–2158.