PRACTICAL USES OF EXISTENTIAL RULES IN KNOWLEDGE REPRESENTATION

Part 4: Practical Applications of Rules

David Carral,[1] Markus Krötzsch,[1] and Jacopo Urbani[2]

1. TU Dresden
2. Vrije Universiteit Amsterdam

ECAI, September 4, 2020

Outline

Goal

Show some example where either **rules** or **related ideas** were crucial to achieve the state of the art

- PLP
- Data integration
- Stream reasoning

Take-home message

1. Rules can be used also in scenarios where not everything is definite
2. A **declarative** approach is (often) intuitive and **decreases** the development time
3. Developing robust tools is fundamental

1^{st} Scenario: Probabilistic Logic Programming

PLP

How can we perform logic-based reasoning in an uncertain domain?

PLP

Probabilistic Logic Programming (PLP): Formalisms to combine logic and probability for reasoning in uncertain domains.

Basic idea: Reason over facts which may be true with a certain probability

State of the art

Several PLP formalisms exist. **ProbLog** (Raedt, Kimmig, and Toivonen 2007) is one of the most popular ones

ProbLog

Definition

A ProbLog program $\mathcal{P}$ is a triple $(\mathcal{R}, \mathcal{F}, \pi)$ where $\mathcal{R}$ is set of (function-free) rules, $\mathcal{F}$ is a set of facts and $\pi : \mathcal{F} \rightarrow [0, 1]$ is the function that labels facts with probabilities.

Key problem

Given $\mathcal{P}$ and query q as input, what is $Pr(q)$ (the probability of q)?

General Approach

It has been shown that computing $Pr(q)$ can be expressed using Weighted Model Counting (WMC) over weighted logical formulas (Vlasselaer et al. 2016)

The Grounding Problem

ProbLog2, a state-of-the-art engine, proceeds as follows:

1. Find relevant **ground** program for q with backward chaining
2. Execute a custom implementation of fixpoint operator $T_{\mathcal{P}}$:
   - $T_{\mathcal{P}}$ proceeds bottom-up, akin to chase procedures
   - $T_{\mathcal{P}}$ incrementally computes, for each inferred fact f, a propositional formula λ_f which "remembers" all the possible ways f can be inferred
3. After $T_{\mathcal{P}}$ has finished, it computes WMC for λ_q

Problem

Grounding can be a major performance bottleneck with large knowledge bases

Datalog to the rescue

Some ideas developed for Datalog are useful here (Tsamoura, Gutiérrez-Basulto, and Kimmig 2020)

First idea

Don't ground $\mathcal{P}$ with backward chaining. Rewrite it with **magic sets** (Bancilhon et al. 1985)

Second idea

Apply **semi-naïve evaluation** (Abiteboul, Hull, and Vianu 1995) while computing $T_{\mathcal{P}}$ to reduce the number of duplicates

Magic sets

Consider database I and program P. Our goal is to answer query Q

Idea

The main idea is to rewrite P into P' where additional **magic** relations restrict the derivations to facts relevant for answering Q

Magic sets

Consider database I and program P. Our goal is to answer query Q

Example 1

Consider the rules below and assume we want to answer $Q = lives(linda, X)$

$$married(X, Y), lives(X, Z) \rightarrow lives(Y, Z) \qquad (r_1)$$

$$married(X, Y) \rightarrow married(Y, X) \qquad (r_2)$$

The rewriting procedure produces the program

$$mgc_1(Y), married(X, Y), lives(X, Z) \rightarrow lives(Y, Z) \qquad (r_3)$$

$$mgc_1(X) \rightarrow mgc_2(X) \qquad (r_4)$$

$$mgc_2(Y), married(X, Y) \rightarrow married(Y, X) \qquad (r_5)$$

Then, we can reason on $I \cup \{mgc_1(linda)\}$

Semi naïve evaluation

Semi naïve evaluation is a well-known technique to avoid the recomputation of duplicate derivation during the materialization

Naïve Evaluation

Input: Facts I, program P

while true do
  $J := I$;
  for $r \in P$ **do**
    Let r be $B \to H$
    $J := J \cup \{H\sigma \mid B\sigma \subseteq I\}$;
  if $J = I$ **then return** J ;

Semi Naïve Evaluation

Input: Facts I, program P

$\Delta := I$;
while true do
  $J := I$;
  for $r \in P$ **do**
    Let r be $B \to H$;
    $J := J \cup \{H\sigma \mid B\sigma \subseteq I \wedge B\sigma \cap \Delta \neq \emptyset\}$;
  if $J = I$ **then return** J;
  $\Delta := J \setminus I$;

New approach

Tsamoura et al. (2020) proposed a new procedure:

1. ~~Find relevant **ground** program for q with backward chaining~~. Use Magic Set to obtain a **non-ground** program
2. ~~Execute a custom implementation of fixpoint operator $T_{\mathcal{P}}$~~ Offload the computation to a chase engine (VLog):
   - Leverage semi-naïve evaluation
   - Introduce some rules to compute formulas (called λ–transformation)
3. After $T_{\mathcal{P}}$ has finished, compute WMC for λ_q

Impact

The new procedure removes the need for grounding, which was a performance bottleneck

Performance improvement

Some key results from (Tsamoura, Gutiérrez-Basulto, and Kimmig 2020)

- The runtime of query answering was two order of magnitude and 25% faster than ProbLog2 in the best and worst cases, respectively
- VLog enabled the computation on much larger DBs than what was possible before

Lesson learned

Well-known ideas developed for rule-based query answering can be re-used as-is for other problems as well

2^{nd} Scenario: Entity Resolution

Problem

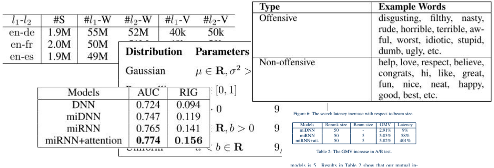
Scientific advancement requires an extensive analysis of prior knowledge in the literature, but this is **time consuming**

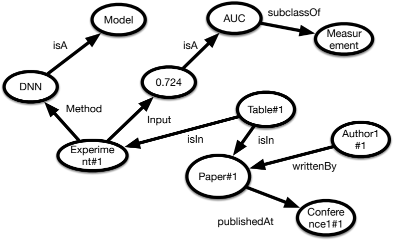
AI can help!

Long-term vision: Develop an accurate and large-scale KB of scientific knowledge

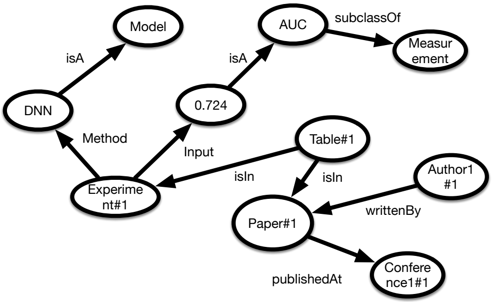
A KB of Scientific Knowledge

valuable experimental knowledge





Advantages



Potential use cases:

- Retrieve experimental results with entity-based search
- Exploit co-authorship networks
- Identify potential inconsistencies across papers

Tab2Know: General pipeline

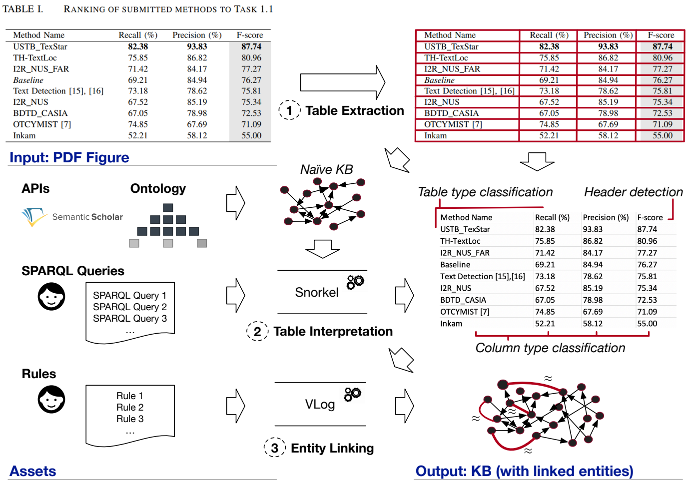
Tab2Know is a recent work to construct a KB from tables in scientific papers (Kruit, He, and Urbani 2020)

Key features:

- Heuristic-based methods to recognize and extract tables from PDFs
- Machine learning models to predict the type of tables and columns
- **Weak supervision** with SPARQL queries to counter the problem of lack of training data
- **(Focus of today)** logic-based reasoning for **entity resolution**

Tab2Know: General pipeline

From (Kruit, He, and Urbani 2020)



Entity Resolution

Entity resolution is the task of recognizing and linking entities across different tables. It is a well-known task in database literature (96+ papers between 2009-2014, see (Papadakis, Ioannou, and Palpanas 2020))

- Magellan (Konda et al. 2016)
- Deep Learning (Mudgal et al. 2018)
- Crowd-sourcing (Das et al. 2017)
- Embeddings (Cappuzzo, Papotti, and Thirumuruganathan 2020)
- ...

A declarative approach

Tab2Know's approach: Use (existential) rules!

TGDs

Used to create new entities from the cells

EGDs

Used to infer equality among the entities

Output

After reasoning is completed, entities are used to populate a KB

A declarative approach: TGDs

Two TGDs are used:

$$type(X, \text{Column}) \rightarrow \exists Y.colEntity(X, Y) \quad (r_1)$$
$$type(X, \text{Cell}) \rightarrow \exists Y.cellEntity(X, Y) \quad (r_2)$$

- Two types of entities are introduced. One describes columns, the other describes cells;
- Every cell is assigned to a entity; it is likely that the same entity is represented with multiple labeled nulls!

A declarative approach: EGDs

EGDs determines whether multiple cells refer to the same entity

$$ceNoTypLabel(X,L) \wedge ceNoTypLabel(Y,L) \rightarrow X \approx Y \qquad (r_3)$$

$$eNoTypLabel(X,C,L), eNoTypLabel(Y,C,L) \rightarrow X \approx Y \qquad (r_4)$$

$$eTableLabel(X,T,L), eTableLabel(Y,T,L) \rightarrow X \approx Y \qquad (r_5)$$

$$eTypLabel(X,S,L), eTypLabel(Y,S,M), STR\_EQ(L,M) \rightarrow X \approx Y \qquad (r_6)$$

$$eAuthLabel(X,A,L), eAuthLabel(Y,A,M), STR\_EQ(L,M) \rightarrow X \approx Y \qquad (r_7)$$

- Special built-in predicates (STR_EQ) encode string similarities
- Other predicates include authors of the paper
- Program can be easily extended with other rules → rapid KB construction

Preliminary results

Input

Approach was tested on a collection with 142k CS open-access papers and 73k tables (IJCAI, ECAI, etc.)

Key results

- Table interpretation superior than previous state-of-the-art approach (Yu et al. 2020)
- EGDs reduced number of “column” entities of 65% and of “cell” entities of 55%
- Every rule contributed by linking some entities
- On a sample of 541 entities, average precision was 97%

Lessons learned

1. A declarative approach is ideal for non-CS domain experts
2. Rules can be easily changed or adapted depending on the performance
3. VLog was scalable enough to perform rapid prototyping with large KGs
4. Support to built-in predicates was crucial

3^{rd} Scenario: Stream Reasoning

A few of slides are a modified version of Harald Beck's ISWC17 presentation, used with permission

Motivation

Stream reasoning: add reasoning on top of stream processing. Central question: **"What is true now?"** (Margara et al. 2014)

- E.g. public transport: What are the current expected arrival times?
- Is there currently a good connection between two lines?

Semantic Web: RDF Stream Processing - SPARQL extensions: C-SPARQL, CQELS, $\text{SPARQL}_{\text{Stream}}$, ... Typical: Window operators select snapshots of recent data

- Window examples: `[RANGE 3m]`, `[TRIPLES 2]`

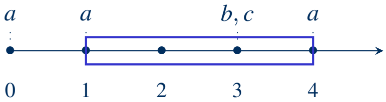
Goals & Challenges

- Goal: expressive stream reasoning solutions
  (1) based on model-based semantics
  (2) high performance

- Central challenge: **throughput vs. expressiveness**

LARS: A **L**ogic for **A**nalytic **R**easoning over **S**treams

LARS (Beck, Dao-Tran, and Eiter 2018) is a logic-based frameworks to reason on streams



- Stream $S = (T, \upsilon)$
  - **Timeline** T closed interval in $\mathbb{N}$, $t \in T$ **time point**
  - **Evaluation** function $\upsilon : T \to 2^{\mathcal{A}}$ (sets of atoms)
- Window function w yields window $w(S, t) \subseteq S$
- Formulas ψ: evaluated on S at t

ψ	holds in S at t iff φ holds ...	Ex.: $S, 4 \models \psi$?
$\boxplus^w \varphi$	in $w(S, t)$ at t	
$\Diamond \varphi$	at **some** time point $t' \in T$	$\boxplus^3 \Diamond a$ ✓
$\Box \varphi$	at **all** time points $t' \in T$	$\boxplus^3 \Box a$ x
$@_{t'} \varphi$	at time point t' and $t' \in T$	$\boxplus^3 @_1 a$ ✓

Plain LARS

Observations

- Many practical problems do not need a multiple model semantics
- **Time-based** and **tuple-based windows** often suffice
- Sliding windows can be exploited for incremental reasoning

Plain LARS (Bazoobandi, Beck, and Urbani 2017)

Focus on positive LARS programs where for each rule $\alpha \leftarrow \beta_1, \ldots, \beta_n$ we have:

- head α: atom a or $@_t\, a$
- body elements: $\beta_i ::= a \mid @_t\, a \mid \boxplus^w @_t\, a \mid \boxplus^w \Diamond\, a \mid \boxplus^w \Box\, a$

Consider **non-ground programs**, using substitutions due to available ground atoms, as usual

From LARS to Datalog

Observation

LARS rules can be rewritten into Datalog rules

- How do we represent time?
  - Increase arity of the relations, e.g., $P(X) \rightarrow P(X, T)$
- How can we translate LARS rules?
  - $@_S P(X)$ as $P(X, S)$
  - $\boxplus^2 \Diamond P(X) \rightarrow Q(X)$ as $P(X, T) \rightarrow Q(X)$ and $P(X, T-1) \rightarrow Q(X)$

Semi-naïve evaluation (SNE)

One key novelty of (Bazoobandi, Beck, and Urbani 2017) is to show how to replicate SNE in a stream

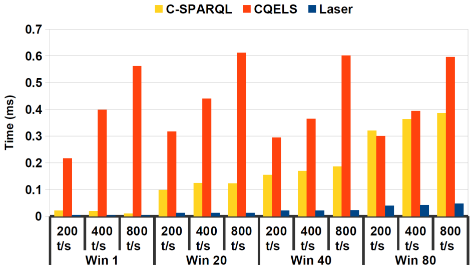
From LARS to Datalog

- For formula $\varphi = \alpha, \beta_i$ in any rule $\alpha \leftarrow \beta_1, \ldots, \beta_n$, consider **annotated ground formulas** $\varphi\sigma_{[c,h]}$, where
  - $\varphi\sigma$ is the **ground instance** of φ due to **substitution** σ
  - $[c, h]$ is an annotation stating that $\varphi\sigma$ holds from **consideration time** c to **horizon time** h
- Horizon time can be extended in the future, e.g., at time point t, $\boxplus^3 \Diamond p(a)$ can be annotated as $\boxplus^3 \Diamond p(a)_{[t,t+3]}$
- When computing substitution σ for instantiating rule $B_1 \wedge B_2 \wedge \ldots B_n \rightarrow H$ at time point t, at least one $B_i\sigma_{[c,h]}$ has $c = t$, i.e., has been derived at the current time point

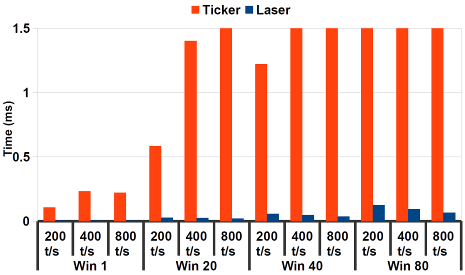
Laser: Implementation & Evaluation

Evaluation: Time per triple

- Compare to C-SPARQL, CQELS, and Ticker
- Micro benchmarks to test **(1)** $q(A, B) \leftarrow \boxplus^n \Diamond p(A, B)$ (resp. $\Box$); elementary data join; multiple rules; **(2)** small show case example requiring LARS features.
- Window sizes: 1s to 80s; stream rate: 200 to 800 triples/second



(1)



(2)

Lesson learned

- A good idea remains a good idea (even if is old)
- ... but it might need to be properly implemented

To conclude

We have described cases where rules turned out to be very useful

- In some scenarios, existential quantification was necessary (data integration). In others, Datalog rules were enough (PLP, stream reasoning)
- Sometimes, the tools could be directly used (data integration). In other cases, some modifications are required (PLP)
- Finally, we have seen how sometimes **ideas** rather than technology can make the difference

References I

- Abiteboul, Serge, Richard Hull, and Victor Vianu (1995). **Foundations of databases**. Vol. 8. Addison-Wesley Reading.
- Bancilhon, Francois, David Maier, Yehoshua Sagiv, and Jeffrey D. Ullman (1985). “Magic sets and other strange ways to implement logic programs”. In: **Proceedings of the fifth ACM SIGACT-SIGMOD symposium on Principles of database systems**. ACM, pp. 1–15. (Visited on 02/25/2015).
- Bazoobandi, Hamid R., Harald Beck, and Jacopo Urbani (2017). “Expressive Stream Reasoning with Laser”. In: **ISWC**, pp. 87–103.
- Beck, Harald, Minh Dao-Tran, and Thomas Eiter (2018). “LARS: A Logic-based framework for Analytic Reasoning over Streams”. In: **Artificial Intelligence** 261, pp. 16–70. ISSN: 0004-3702.
- Cappuzzo, Riccardo, Paolo Papotti, and Saravanan Thirumuruganathan (2020). “Creating Embeddings of Heterogeneous Relational Datasets for Data Integration Tasks”. In: **SIGMOD**, pp. 1335–1349.

References II

- Das, Sanjib, Paul Suganthan G.C., AnHai Doan, Jeffrey F. Naughton, Ganesh Krishnan, Rohit Deep, Esteban Arcaute, Vijay Raghavendra, and Youngchoon Park (2017). "Falcon: Scaling Up Hands-Off Crowdsourced Entity Matching to Build Cloud Services". In: **SIGMOD**, pp. 1431–1446.
- Konda, Pradap, Sanjib Das, Paul Suganthan G. C., AnHai Doan, Adel Ardalan, Jeffrey R. Ballard, Han Li, Fatemah Panahi, Haojun Zhang, Jeff Naughton, Shishir Prasad, Ganesh Krishnan, Rohit Deep, and Vijay Raghavendra (2016). "Magellan: toward building entity matching management systems". In: **PVLDB** 9.12, pp. 1197–1208.
- Kruit, Benno, Hongu He, and Jacopo Urbani (2020). "Tab2Know: Building a Knowledge Base from Tables in Scientific Papers". In: **To appear at ISWC 2020**, pp. xxx–xxx.
- Margara, Alessandro, Jacopo Urbani, Frank Van Harmelen, and Henri Bal (2014). "Streaming the web: Reasoning over dynamic data". In: **Web Semantics: Science, Services and Agents on the World Wide Web** 25, pp. 24–44. (Visited on 04/30/2017).

References III

Mudgal, Sidharth, Han Li, Theodoros Rekatsinas, AnHai Doan, Youngchoon Park, Ganesh Krishnan, Rohit Deep, Esteban Arcaute, and Vijay Raghavendra (2018). "Deep Learning for Entity Matching: A Design Space Exploration". In: **SIGMOD**, pp. 19–34.

Papadakis, George, Ekaterini Ioannou, and Themis Palpanas (2020). "Entity Resolution: Past, Present and Yet-to-Come.". In: **EDBT**, pp. 647–650.

Raedt, Luc De, Angelika Kimmig, and Hannu Toivonen (2007). "ProbLog: A Probabilistic Prolog and Its Application in Link Discovery". In: **IJCAI**, pp. 2462–2467.

Tsamoura, Efthymia, Víctor Gutiérrez-Basulto, and Angelika Kimmig (2020). "Beyond the Grounding Bottleneck: Datalog Techniques for Inference in Probabilistic Logic Programs". In: **AAAI**, pp. 10284–10291.

Vlasselaer, Jonas, Guy Van den Broeck, Angelika Kimmig, Wannes Meert, and Luc De Raedt (2016). "TP-Compilation for inference in probabilistic logic programs". In: **International Journal of Approximate Reasoning** 78, pp. 15–32.

References IV

Yu, Wenhao, Wei Peng, Yu Shu, Qingkai Zeng, and Meng Jiang (2020). "Experimental Evidence Extraction System in Data Science with Hybrid Table Features and Ensemble Learning". In: **WWW**, pp. 951–961.