PRACTICAL USES OF EXISTENTIAL RULES IN KNOWLEDGE REPRESENTATION

Part 3: Applications of Rules in AI

David Carral,¹ Markus Krötzsch,¹ and Jacopo Urbani²
1. TU Dresden
2. Vrije Universiteit Amsterdam

KR 2020, September 13, 2020
Goal
Show some example where either rules or related ideas were crucial to achieve the state of the art

- Horn-$\mathcal{ALC}$ reasoning
- PLP
- Data integration
- Stream reasoning
Goal
Show some example where either rules or related ideas were crucial to achieve the state of the art

- Horn-\(\mathcal{ALC}\) reasoning
- PLP
- Data integration
- Stream reasoning
Take-home message

1. Rules can be used also in uncertain scenarios
2. A declarative approach is (often) intuitive and decreases the development time
3. Robust and scalable reasoning tools are crucial
4. AI communities should talk to each other!
2\textsuperscript{nd} Scenario: Probabilistic Logic Programming
How can we perform logic-based reasoning in an uncertain domain?
How can we perform logic-based reasoning in an uncertain domain?

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

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
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 $T_P$:
   - $T_P$ proceeds bottom-up, akin to chase procedures
   - $T_P$ incrementally computes, for each inferred fact $f$, a propositional formula $\lambda_f$ which “remembers” all the possible ways $f$ can be inferred
3. After $T_P$ has finished, it computes $WMC$ for $\lambda_q$

Problem

Grounding can be a major performance bottleneck with large knowledge bases
Some ideas developed for Datalog can be useful (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)
Some ideas developed for Datalog can be useful (Tsamoura, Gutiérrez-Basulto, and Kimmig 2020)

**First idea**
Don’t ground $P$ with backward chaining. Rewrite it with **magic sets** (Bancilhon et al. 1985)

**Second idea**
Apply **semi-naïve evaluation** (Abiteboul, Hull, and Vianu 1995) on the non-ground program to reduce the number of duplicates
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 $Q$. 
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 \( Q = \text{lives(linda, X)} \)

\[
\begin{align*}
\text{married}(X, Y), \text{lives}(X, Z) & \rightarrow \text{lives}(Y, Z) \\
\text{married}(X, Y) & \rightarrow \text{married}(Y, X)
\end{align*}
\]

\((r_1)\) \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{
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 $Q = \text{lives}(\text{linda}, X)$

$$\text{married}(X, Y), \text{lives}(X, Z) \rightarrow \text{lives}(Y, Z) \quad (r_1)$$

$$\text{married}(X, Y) \rightarrow \text{married}(Y, X) \quad (r_2)$$

The rewriting procedure produces the program

$$\text{mgc}_1(Y), \text{married}(X, Y), \text{lives}(X, Z) \rightarrow \text{lives}(Y, Z) \quad (r_3)$$

$$\text{mgc}_1(X) \rightarrow \text{mgc}_2(X) \quad (r_4)$$

$$\text{mgc}_2(Y), \text{married}(X, Y) \rightarrow \text{married}(Y, X) \quad (r_5)$$

Then, we can reason on $I \cup \{\text{mgc}_1(\text{linda})\}$
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 $B \rightarrow H$
        $J := J \cup \{H\sigma \mid B\sigma \subseteq I\}$;
    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 $B \rightarrow H$
        $J := J \cup \{H\sigma \mid B\sigma \subseteq I \land B\sigma \cap \Delta \neq \emptyset\}$;
    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.
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 $T_p$. Offload the computation to a chase engine (VLog):
   - Leverage semi-naïve evaluation
   - Introduce extra rules to compute the probability ($\lambda$-transformation)

Impact

The new procedure removes the need for grounding, which was a performance bottleneck.
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 $T_P$ Offload the computation to a chase engine (VLog):
   - Leverage semi-naïve evaluation
   - Introduce extra rules to compute the probability ($\lambda$-transformation)

3. After $T_P$ has finished, compute $WMC$ for $\lambda_q$

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
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
3rd Scenario: 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)
- …
Scientific advancement requires an extensive analysis of prior knowledge in the literature, but this is **time consuming**
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
cost of our models, we compare the online search latency of different models. The latency of the baseline is 21 ms. And the relative latency increase of our models over the baseline are shown in Figure 4.

In Figure 4, the latency of miDNN is small and grows linearly with respect to rerank size. But the latency of miRNN and miRNN+attention grows polynomially. When rerank size is 50, the latency of miRNN+attention increases 400% over the baseline, from 21 ms to 105 ms. Although the RNN models achieve larger GMV, the computational cost of the RNN models are huge when rerank size gets big. The large computational cost is the major drawback of our RNN models.

For RNN models, we use beam search to find a good ranking sequence. The beam size $k$ is a key parameter for beam search. Larger $k$ means larger search space and usually results in better ranking results. But larger $k$ also lead to more computational cost. We studied the GMV and latency increase with respect to beam size. And the results are shown in Figure 5 and Figure 6.

Figure 5 shows that the GMV increases as beam size grows. But the GMV increase gets smaller when beam size gets larger. Figure 6 shows that the latency increases linearly with respect to beam size, which is in accordance with our time complexity analysis. A balance of GMV and latency is needed to choose the value of beam size. And we set the beam size to 5.

Finally, we summarize our online test results in Table 2. The rerank size is set to 50 and the beam size for RNN models is 5. Results in Table 2 show that our mutual influence aware ranking framework brings a significant GMV increase over the baseline. The miDNN model achieves a good GMV increase with only a little latency overhead. The miRNN+attention model gets the best GMV result but the latency grows too fast. The miRNN model achieves a very good GMV increase with much less latency compared to miRNN+attention. Therefore, if computational cost is very expensive, the miDNN model is a good choice. In our case where mild latency increase is acceptable, the miRNN model is preferred.

5 Conclusion

In this paper, we point out the importance of mutual influences between items in e-commerce ranking and propose a global optimization framework for mutual influence aware ranking for the first time. We incorporate mutual influences into our models by global feature extension and modeling ranking as a sequence generation problem. We performed online experiments on a large e-commerce search engine. To reduce computational cost, we use our methods as a reranking process on top of the baseline ranking. The results show that our method produces a significant GMV increase over the baseline, and therefore verifies the importance of mutual influences between items. We also compared the computational costs of our methods. Our miDNN model noticeably increases GMV without much computational cost. Our attention mechanism for RNN model gets the best GMV result. But the computational cost of our attention mechanism is too high. Future research will be focused on more efficient attention mechanisms that increase GMV with less computations.

Acknowledgments

This work receives great help from our colleague Xiaoyi Zeng. We would also like to thank Xin Li and the Taobao Search Engineering team for helpful discussions and the system engineering efforts.
A KB of Scientific Knowledge

valuable experimental knowledge

<table>
<thead>
<tr>
<th>Type</th>
<th>Example Words</th>
</tr>
</thead>
<tbody>
<tr>
<td>Offensive</td>
<td>disgusting, filthy, nasty, rude, horrible, terrible, awful, worst, idiotic, stupid, dumb, ugly, etc.</td>
</tr>
<tr>
<td>Non-offensive</td>
<td>help, love, respect, believe, congrats, hi, like, great, fun, nice, neat, happy, good, best, etc.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Models</th>
<th>AUC</th>
<th>RIG</th>
<th>$\mu \in \mathbb{R}$, $\sigma^2 &gt; 0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNN</td>
<td>0.724</td>
<td>0.094</td>
<td></td>
</tr>
<tr>
<td>miDNN</td>
<td>0.747</td>
<td>0.119</td>
<td></td>
</tr>
<tr>
<td>miRNN+attention</td>
<td>0.774</td>
<td>0.156</td>
<td></td>
</tr>
</tbody>
</table>

Figure 6: The search latency increase with respect to beam size.

Table 2: The GMV increase in A/B test.

<table>
<thead>
<tr>
<th>Models</th>
<th>Rerank size</th>
<th>Beam size</th>
<th>GMV</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>miDNN</td>
<td>50</td>
<td>5</td>
<td>2.91%</td>
<td>9%</td>
</tr>
<tr>
<td>miRNN</td>
<td>50</td>
<td>5</td>
<td>5.03%</td>
<td>58%</td>
</tr>
<tr>
<td>miRNN+attention</td>
<td>50</td>
<td>5</td>
<td>5.82%</td>
<td>401%</td>
</tr>
</tbody>
</table>

5 Conclusion

In this paper, we point out the importance of mutual influences between items in e-commerce ranking and propose a global optimization framework for mutual influence aware ranking for the first time. We incorporate mutual influences into our models by global feature extension and modeling ranking as a sequence generation problem. We performed online experiments on a large e-commerce search engine. To reduce computational cost, we use our methods as a reranking process on top of the baseline ranking. The results show that our method produces a significant GMV increase over the baseline, and therefore verifies the importance of mutual influences between items. We also compared the computational costs of our methods. Our miDNN model noticeably increases GMV without much computational cost. Our attention mechanism for RNN model gets the best GMV result. But the computational cost of our attention mechanism is too high. Future research will be focused on more efficient attention mechanisms that increase GMV with less computations.

Acknowledgments

This work receives great help from our colleague Xiaoyi Zeng. We would also like to thank Xin Li and the Taobao Search Engineering team for helpful discussions and the system engineering efforts.
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
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
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
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**
Fig. 1.1: Results table from ICDAR 2013 task. Tables, on the other hand, provide direct structured data which is easily ingestable into a knowledge base. Take the table in figure 1.1 taken from the paper presenting the results for the ICDAR 2013 document recognition competition (1). The information density in this table is incredible. We see multiple method names, some of which are even directly linked to citations, with their respective recall, precision and F-score. If the structure of this table is known, structured data could easily be extracted. This data were to be of great use for the augmentation of existing knowledge bases. An example relationship that could be extracted would be a method and its score. The caption “Ranking of submitted methods to task 1.1” also gives us additional insights. We know that task 1.1 is probably defined somewhere else in the document, and linking this to the performance of a specific method is of great value. A simple search in the document provides us with the information that task 1.1 is “Text Localization”. Thus we know:

\[
\text{USTB_TexStar} \quad \text{performs with} \quad 87.75 \text{ F-score} \quad \text{on task} \quad \text{Text Localization}
\]

Linking this information and making it available through a search engine, would greatly enhance the experience of researchers.

Another useful application of table data from academic papers would be as follows: imagine a direct query that provides a researcher with all the papers that report an exact same technique for the exact same task. This could even be provided to the user in table mark-up and could clearly show the discrepancies between similar research. Many computer science experiments are highly reproducible, but will never be questioned since such contradictory information is never found. This task does not necessarily question the professionalism and integrity of researchers, but could definitely give us great insights.

---

### Tab2Know: General pipeline

From (Kruit, He, and Urbani 2020)

**Input: PDF Figure**

- **APIs**
  - Semantic Scholar

- **Ontology**

- **SPARQL Queries**
  - SPARQL Query 1
  - SPARQL Query 2
  - SPARQL Query 3

- **Rules**
  - Rule 1
  - Rule 2
  - Rule 3

- **Assets**

**Output: KB (with linked entities)**

1. **Table Extraction**
2. **Table Interpretation**
3. **Entity Linking**

**Header detection**

**Table type classification**

**Column type classification**

**SPARQL Queries**

- SPARQL Query 1
- SPARQL Query 2
- SPARQL Query 3

**Rules**

- Rule 1
- Rule 2
- Rule 3

**Assets**

---

J. Urbani, September 13, 2020

Practical Uses of Existential Rules in Knowledge Representation
Table Extraction

Table Interpretation

Entity Linking

Output: KB (with linked entities)
A declarative approach

**Terminology**

Tuple Generating Dependency (TGD): $bicycle(X) \rightarrow \exists Y. partOf(X, Y) \land Wheel(Y)$

Equality Generating Dependency (EGD): $email(X, Y) \land email(X, Z) \rightarrow Y \approx Z$

Tab2Know's approach: Use TGDs and EGDs to perform entity resolution

**TGDs**

They can be used to create new entities from the cells and columns

**EGDs**

They can be used to infer that entities mentioned in different cells are the same

**Output**

After reasoning is completed, newly introduced entities are used to populate a KB
A declarative approach: TGDs

Two types of entities: One for columns, one for cells

\[ type(X,\, \text{Column}) \rightarrow \exists Y.\text{colEntity}(X,\, Y) \]  \hspace{1cm} (r_1)

\[ type(X,\, \text{Cell}) \rightarrow \exists Y.\text{cellEntity}(X,\, Y) \]  \hspace{1cm} (r_2)
A declarative approach: EGDs

Avoid that the same entity is represented with multiple labeled nulls

\[ ceNoTypLabel(X, L) \land ceNoTypLabel(Y, L) \rightarrow X \approx Y \]  
\[ eNoTypLabel(X, C, L), eNoTypLabel(Y, C, L) \rightarrow X \approx Y \]  
\[ eTableLabel(X, T, L), eTableLabel(Y, T, L) \rightarrow X \approx Y \]  
\[ eTypLabel(X, S, L), eTypLabel(Y, S, M), STR_EQ(L, M) \rightarrow X \approx Y \]  
\[ eAuthLabel(X, A, L), eAuthLabel(Y, A, M), STR_EQ(L, M) \rightarrow X \approx Y \]

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

Input
Approach was tested on a collection with 142k CS open-access papers and 73k tables (IJCAI, ECAI, etc.)
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%
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
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
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
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
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
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?
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, SPARQLStream, ... 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 Logic for Analytic Reasoning over Streams

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

\[
\begin{array}{cccc}
  a & a & b, c & a \\
  \vdots & \vdots & \vdots & \vdots \\
  0 & 1 & 2 & 3 \\
\end{array}
\]

- **Stream** \( S = (T, \nu) \)
  - **Timeline** \( T \) closed interval in \( \mathbb{N} \), \( t \in T \) **time point**
  - **Evaluation** function \( \nu : T \to 2^A \) (sets of atoms)
LARS: A Logic for Analytic Reasoning over Streams

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

\[
\text{Stream } S = (T, \nu)
\]

- **Timeline** $T$ closed interval in $\mathbb{N}$, $t \in T$ **time point**
- **Evaluation** function $\nu : T \rightarrow 2^A$ (sets of atoms)

**Formulas** $\alpha$: evaluated on $S$ at $t$

- $\alpha = \forall^w \beta$: means that $\beta$ must hold on the substream defined by a window function with arg $w$ (e.g., last $w$ time points)
- $\alpha = \diamond \beta$: means that $\beta$ must hold at **some** time point
LARS: A Logic for Analytic Reasoning over Streams

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

Stream $S = (T, \nu)$

- **Timeline** $T$ closed interval in $\mathbb{N}$, $t \in T$ time point
- **Evaluation** function $\nu: T \rightarrow 2^A$ (sets of atoms)

Formulas $\alpha$: evaluated on $S$ at $t$

- $\alpha = \Box^w \beta$: means that $\beta$ must hold on the substream defined by a window function with arg $w$ (e.g., last $w$ time points)
- $\alpha = \Diamond \beta$: means that $\beta$ must holds at some time point
- $\alpha = \square \beta$: means that $\beta$ must holds at every time point
LARS: A Logic for Analytic Reasoning over Streams

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

- **Stream** $S = (T, \nu)$
  - **Timeline** $T$ closed interval in $\mathbb{N}$, $t \in T$ time point
  - **Evaluation** function $\nu : T \to 2^A$ (sets of atoms)

- **Formulas** $\alpha$: evaluated on $S$ at $t$
  - $\alpha = \bigoplus^w \beta$: means that $\beta$ must hold on the substream defined by a window function with arg $w$ (e.g., last $w$ time points)
  - $\alpha = \Diamond \beta$: means that $\beta$ must hold at some time point
  - $\alpha = \square \beta$: means that $\beta$ must hold at every time point
  - $\alpha = @T \beta$: means that $\beta$ must hold at time point $T$
Plain LARS (Bazoobandi, Beck, and Urbani 2017)

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

- **head $\alpha$:** atom $a$ or $\@_ta$
- **body elements:** $\beta_i ::= a \mid \@_ta \mid \mathbb{W}\@_ta \mid \mathbb{W}a \mid \mathbb{W}\Box a$
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 \( P(X, S) \)
  - \( \square^2 \Box P(X) \rightarrow Q(X) \) as \( P(X, T) \rightarrow Q(X) \) and \( P(X, T - 1) \rightarrow Q(X) \)
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 $P(X, S)$
  - $\bowtie^2 \diamond P(X) \rightarrow Q(X)$ 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
• 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 \( \varphi \) due to **substitution** \( \sigma \)
  
  – \( [c, h] \) is an **annotation** stating that \( \varphi\sigma \) holds from **consideration time** \( c \) to **horizon time** \( h \)
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 $\varphi$ due to substitution $\sigma$
- $[c, h]$ 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 $t$, $\Box^3 p(a)$ can be annotated as $\Box^3 p(a)_{[t, t+3]}$
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 $\varphi$ due to **substitution** $\sigma$
- $[c, h]$ 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 $t$, $\Box^3 \Diamond p(a)$ can be annotated as $\Box^3 \Diamond p(a)_{[t, t+3]}$

When computing substitution $\sigma$ for instantiating rule $B_1 \land B_2 \land \ldots B_n \rightarrow H$ 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) $q(A, B) \leftarrow p^n \diamond p(A, B)$ (resp. □); 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
Lesson learned

- A good idea remains a good idea (even if is old)
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
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 systems, can make the difference


