Reasoning at Scale: Why, How and What’s Next.

Efi Tsamoura
Samsung AI, Cambridge, UK
# Datalog Reasoning with Trigger Graphs

Table: Reasoning over LUBM for 1B–17B of database triples.

<table>
<thead>
<tr>
<th>#IDPs</th>
<th>1B</th>
<th>2B</th>
<th>4B</th>
<th>8B</th>
<th>17B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Runtime (s)</td>
<td>203</td>
<td>226</td>
<td>520</td>
<td>993</td>
<td>2272</td>
</tr>
<tr>
<td>Memory (GB)</td>
<td>23</td>
<td>34</td>
<td>49</td>
<td>98</td>
<td>174</td>
</tr>
</tbody>
</table>

June 1, 2023
Probabilistic Datalog Reasoning with Lineage Trigger Graphs

Figure: Time in seconds for goal-driven QA over probabilistic LUBM-100.
Reasoning (at Scale): Why
Why: Data Management

– **Industry applications [21]**
  – Microsoft and Google: search & QA.
  – Facebook: user recommendations.
  – Bosch: autonomous driving.
  – Samsung: healthcare.
  – LogicBlox: analytics.

– **Success stories**
  – RDFox.
  – Vadalog (acquired by Meltwater).
Why: Machine Learning

  – The logical theory encodes prior knowledge– the neural model learns a simpler concepts.

– Train with fewer or even no data, e.g., zero-shot learning [8].

– Train in a weak fashion:
  – NeuroLog [25]: abduction + WMC-based loss [4].
Can we Learn via Weak Supervision Coming from Logic? Yes

(Work in progress)

**Theorem**

If \( G \) is unambiguous and any \( f \in \mathcal{F} \) is \( r \)-bounded, then we have:

\[
\mathcal{R}^{01}(f) \leq O(\mathcal{R}_P^{01}(f; G)^{1/M}) \quad \text{as} \quad \mathcal{R}_P^{01}(f; G) \to 0
\]

Furthermore, suppose \([\mathcal{F}]\) has a finite Natarajan dimension \( d_{[\mathcal{F}]} \) and the function class \( \{(y, s) \mapsto 1\{\sigma'(y) \neq s\} | \sigma' \in \mathcal{G}\} \) has a finite VC-dimension \( d_{\mathcal{G}} \). Then, for any \( \epsilon, \delta \in (0, 1) \), there is a universal constant \( C_4 \) such that with probability at least \( 1 - \delta \), the empirical partial risk minimizer with \( \widehat{\mathcal{R}}_P^{01}(f; \sigma) = 0 \) has a classification risk \( \mathcal{R}^{01}(f) < \epsilon \), if

\[
m_P \geq C_4 \frac{c^{2M-2}}{r_M \epsilon^M} \left( ((d_{[\mathcal{F}]} + d_{\mathcal{G}}) \log(6M(d_{[\mathcal{F}]} + d_{\mathcal{G}})) + d_{[\mathcal{F}]} \log c) \log \left( \frac{c^{2M-2}}{r_M \epsilon^M} \right) + \log \left( \frac{1}{\delta} \right) \right)
\]
Believe in KRR
-My neurosymbolic research
Scene Graph Generation (AAAI 2023)

Task

Logic-Based Regularization

Scene Graph Generation (AAAI 2023)

Figure: Comparison against BGNN [16], KBFN [10] and VCTree [22]. Benchmark: Visual Genome [13].
Scene Graph Generation (AAAI 2023)

Figure: Recall of VCTree [22] on the 28 least frequent predicates: without NGP; with NGP. Benchmark: Visual Genome [13].
Knowledge Distillation into Deep Networks (ICML 2023)

Concordia

- First to support general first-order theories.
- Supports semi-/un-/supervised learning.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inference</td>
<td>( \hat{y} = \text{arg max}_y P_N(Y = y</td>
</tr>
<tr>
<td>Training</td>
<td>( \hat{\theta}<em>{t+1} = \text{arg min}</em>\theta (\ell(\hat{y}_N, y) + KL(P_N, P_L)) )</td>
</tr>
<tr>
<td></td>
<td>( \hat{\lambda}<em>{t+1} = \text{arg max}</em>\lambda \prod_{(x) \in D} P_L(X = x, \lambda_t) )</td>
</tr>
</tbody>
</table>

Video Activity Detection (ICML 2023)

\[ \text{SEQ}(B_1, B_2) \land \text{CLOSE}(B_1, B_2) \rightarrow \text{SAME}(B_1, B_2) \]
\[ \text{DOING}(B_1, A) \land \text{SAME}(B_1, B_2) \rightarrow \text{DOING}(B_2, A) \]

Accuracy over 5 runs

<table>
<thead>
<tr>
<th>Model</th>
<th>Avg (%)</th>
<th>Max (%)</th>
<th>Min (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACD+L [17]</td>
<td>86.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MobileNet</td>
<td>90.07</td>
<td>91.36</td>
<td>89.61</td>
</tr>
<tr>
<td>IARG(MobileNet) [14]</td>
<td>90.18</td>
<td>92.39</td>
<td>87.55</td>
</tr>
<tr>
<td>Concordia(MobileNet, L)</td>
<td>90.73</td>
<td>93.19</td>
<td>89.54</td>
</tr>
<tr>
<td>Inception</td>
<td>89.72</td>
<td>91.83</td>
<td>86.84</td>
</tr>
<tr>
<td>IARG(Inception) [14]</td>
<td>88.88</td>
<td>91.67</td>
<td>85.33</td>
</tr>
<tr>
<td>Concordia(Inception, L)</td>
<td>92.75</td>
<td>93.34</td>
<td>92.31</td>
</tr>
</tbody>
</table>

## Entity Linking (ICML 2023)

**Table:** Results on entity linking.

<table>
<thead>
<tr>
<th>Model</th>
<th>F&lt;sub&gt;1</th>
<th>Acc (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT (sp)</td>
<td>0.88</td>
<td>88.5</td>
</tr>
<tr>
<td>Concordia(BERT) (sm)</td>
<td><strong>0.91</strong></td>
<td><strong>91.4</strong></td>
</tr>
</tbody>
</table>

Visual QA (SIGMOD 2023)

Q(O) ← NAME(herbivore, O)
NAME(N, O) ∧ NAME(N', O) → ISA(N', N)
→ ISA(giraffe, herbivore)
→ ISA(deer, herbivore)

Table: Recall@5 on VQAR [11].

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>C5</td>
<td>64.05%</td>
<td>74.62%</td>
<td>87.01%</td>
</tr>
<tr>
<td>C6</td>
<td>56.51%</td>
<td>72.04%</td>
<td>85.45%</td>
</tr>
</tbody>
</table>

How this Reasoning Journey Started
Benchmarking the Chase (PODS 2017)

- **Tasks**
  - Materialization.
  - Query answering.

- **(Some) Engines**
  - RDFox [20].
  - DLV [15].
  - E [23].
  - Graal [1].
  - Pegasus [19].

Benchmarking the Chase (PODS 2017)

– Paper takeaways
  – Equality is challenging.
  – Dictionary encoding played a role.
  – Chase engines could support “realistic” scenarios.

– Practitioners’ takeaways
  – Chase engines were struggling with ~100M facts and few hundreds of rules.
  – LUBM-1k was only supported by one engine running on multiple cores.

How this Journey Started (cont’): ProbLog

  - Support Web-crawled KBs.
  - Reasoning over deep neural classifiers.
  - Clean semantics.

- State of affairs
  - Limited applicability.
  - Could not support LUBM-1.

- Contribution
  - Datalog techniques + provenance semirings.
  - Improved scalability by 100x.

Reasoning at Scale: How -Trigger Graphs

Trigger Graphs: Why

- Key to support goal-driven QA over transitive rules.

- **Standard bottom-up evaluation:**
  - may derive logically redundant facts;
  - may try to execute rules that derive no facts.

- **The above negatively impact the runtime and the memory.**
How: Trigger Graphs

Rules

$$r(X, Y) \rightarrow R(X, Y) \quad (r_1)$$

$$R(X, Y) \rightarrow T(Y, X, Y) \quad (r_2)$$

$$T(Y, X, Y) \rightarrow R(X, Y) \quad (r_3)$$

$$r(X, Y) \rightarrow \exists Z.T(Y, X, Z) \quad (r_4)$$

Facts

$$\rightarrow r(c_1, c_2)$$

Bottom-Up evaluation

$$r(c_1, c_2)$$ $$(r_1)$$

$$T(c_2, c_1, n_1)$$ $$(r_4)$$

$$R(c_1, c_2)$$ $$(r_2)$$

$$T(c_2, c_1, n_1)$$ $$(r_3)$$

$$\emptyset$$ $$(r_4)$$

$$T(c_2, c_1, c_2)$$ $$(r_3)$$

$$R(c_1, c_2)$$ $$(r_2)$$
How: Trigger Graphs

Rules

\[ r(X, Y) \rightarrow R(X, Y) \]  
\[ R(X, Y) \rightarrow T(Y, X, Y) \]  
\[ T(Y, X, Y) \rightarrow R(X, Y) \]  
\[ r(X, Y) \rightarrow \exists Z. T(Y, X, Z) \]  

Facts

\[ \rightarrow r(c_1, c_2) \]
How: Trigger Graphs

Rules

\[ r(X, Y) \rightarrow R(X, Y) \]  \( (r_1) \)

\[ R(X, Y) \rightarrow T(Y, X, Y) \]  \( (r_2) \)

\[ T(Y, X, Y) \rightarrow R(X, Y) \]  \( (r_3) \)

\[ r(X, Y) \rightarrow \exists Z. T(Y, X, Z) \]  \( (r_4) \)

Facts

\[ \rightarrow r(c_1, c_2) \]

Bottom-Up evaluation

\[ \begin{align*}
& \phantom{=} r(X_1, X_2) \\
& \phantom{=} r(X_1, X_2) \\
& \phantom{=} T(X_2, X_1, Z) \\
& \phantom{=} T(X_2, X_1, X_2) \\
& \phantom{=} R(X_1, X_2) \\
& \phantom{=} \emptyset
\end{align*} \]
How: Trigger Graphs

Rules

\[ r(X, Y) \rightarrow R(X, Y) \quad (r_1) \]
\[ R(X, Y) \rightarrow T(Y, X, Y) \quad (r_2) \]
\[ T(Y, X, Y) \rightarrow R(X, Y) \quad (r_3) \]
\[ r(X, Y) \rightarrow \exists Z. T(Y, X, Z) \quad (r_4) \]

Facts

\[ \rightarrow r(c_1, c_2) \]
Trigger graph-based reasoning

TGs delineate the rule executions

- Execute $r_1$ over the input instance.
- Execute $r_2$ over the derivations of $r_1$.
- No other operation is taking place.

Important to node

- Facts are stored inside the nodes, i.e., not stored in a single set like in all bottom-up engines.
- This data separation makes joins run faster.
Trigger graph-based reasoning

Rules

\[ r(X, Y) \rightarrow A(X) \quad (r_1) \]
\[ r(X, Y) \rightarrow A(Y) \quad (r_2) \]
\[ A(X) \land s(X, Z) \rightarrow T(Z) \quad (r_3) \]
Trigger Graphs for Linear Rules

– Phase I: Static TG Computation.
  – Compute a representative instance $B^*$, i.e., one that captures all possible rule execution paths.
  – Compute a plan $G$ that mimics the rule execution when reasoning over $B^*$.

– Phase II: Redundancy Elimination.
  – Eliminate nodes that lead to redundant facts (via detecting preserving homomorphisms).

– Phase III: Reasoning.
  – The computed TG can be used to reason over all input instances.
Trigger Graphs for Linear Rules: Complexity

Let $P$ be a linear program that admits a finite universal model.

Theorem (Complexity)

*Computing a TG for $P$ is double exponential in $P$. If the arity of the predicates in $P$ is bounded, the computation time is (single) exponential.*
Reasoning over Linear Rules

Total materialization times in s

<table>
<thead>
<tr>
<th></th>
<th>LUBM-LI</th>
<th>OUBM-LI</th>
<th>DBpedia-LI</th>
<th>Claros-LI</th>
<th>React.-LI</th>
</tr>
</thead>
<tbody>
<tr>
<td>VLog</td>
<td>1</td>
<td>0.2</td>
<td>3.6</td>
<td>5.0</td>
<td>0.4</td>
</tr>
<tr>
<td>RDFox</td>
<td>22</td>
<td>3.1</td>
<td>44</td>
<td>78</td>
<td>72</td>
</tr>
<tr>
<td>X</td>
<td>4</td>
<td>4.1</td>
<td>36</td>
<td>5</td>
<td>12</td>
</tr>
<tr>
<td>TGs</td>
<td>18</td>
<td>0.1</td>
<td>0.8</td>
<td>0.4</td>
<td>0.3</td>
</tr>
</tbody>
</table>

Pick memory in GB

<table>
<thead>
<tr>
<th></th>
<th>LUBM-LI</th>
<th>OUBM-LI</th>
<th>DBpedia-LI</th>
<th>Claros-LI</th>
<th>React.-LI</th>
</tr>
</thead>
<tbody>
<tr>
<td>VLog</td>
<td>1.6</td>
<td>0.2</td>
<td>2.3</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>RDFox</td>
<td>2.3</td>
<td>0.7</td>
<td>3.5</td>
<td>3.7</td>
<td>3.9</td>
</tr>
<tr>
<td>X</td>
<td>1.6</td>
<td>0.2</td>
<td>3.5</td>
<td>2.6</td>
<td>2.5</td>
</tr>
<tr>
<td>TGs</td>
<td>0.4</td>
<td>0.2</td>
<td>0.8</td>
<td>0.4</td>
<td>0.3</td>
</tr>
</tbody>
</table>
Trigger Graphs for Datalog Rules

**TGs for Linear Rules**
- Static TG computation.
- Use the pre-computed TG to reason over *all* instances.
- Redundancy elimination via detecting preserving homomorphisms.

**TGs for Datalog Rules**
- Interleave TG creation with reasoning.
- The computed TG can be used to reason over the given instance only.
- Redundancy elimination via query containment [3].
Trigger Graphs for Datalog Rules: Example

Rules

\[ r(X, Y) \rightarrow S(X, Y, X) \]  (1)

\[ a(X) \land r(X, Y) \rightarrow S(X, X, Y) \]  (2)

\[ S(X, Y, Z) \rightarrow A(X) \]  (3)
Trigger Graphs for Datalog Rules: Example

Trigger Graph

\[ Q(X) = \exists Y. r(X, Y) \]

Query for \( v_3 \)

\[ Q'(X) = \exists Y. a(X) \land r(X, Y) \]

Query for \( v_4 \)
Trigger Graphs for Datalog Rules: Results

Let $P$ be a Datalog program.

Theorem (Soundness)
For a TG $G$ for $P$, $\text{minDatalog}(G)$ is a TG for $P$.

Theorem (Minimality)
Any TG for $P$ has at least as many nodes as $\text{minDatalog}(G)$.

Theorem (Complexity)
Deciding whether $G$ is a TG of minimum size for $P$ is co-NP-complete.
More: TG-Aware Rule Execution Strategy

(i) $a \otimes b = A$

(ii) $a' \otimes b' = A$

(iii) $a' \otimes b' = d$

(iv) $a \otimes b = A$

(v) $a' \triangleright A = d$

(vi) $b' \otimes d = d$
Datalog Reasoning with Trigger Graphs

Materialization times in s

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Materialization Times in s</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUBM-L</td>
<td>16</td>
</tr>
<tr>
<td>LUBM-LE</td>
<td>16</td>
</tr>
<tr>
<td>OUBM-L</td>
<td>2</td>
</tr>
<tr>
<td>DBpedia-L</td>
<td>19</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Pick Memory in GB</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUBM-L</td>
<td>0.3</td>
</tr>
<tr>
<td>LUBM-LE</td>
<td>0.2</td>
</tr>
<tr>
<td>OUBM-L</td>
<td>1.3</td>
</tr>
<tr>
<td>DBpedia-L</td>
<td>0.8</td>
</tr>
</tbody>
</table>

June 1, 2023  ESWC'2023
Datalog Reasoning with Trigger Graphs

Materialization times in minutes

<table>
<thead>
<tr>
<th></th>
<th>VLog</th>
<th>RDFox</th>
<th>X</th>
<th>TGs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Claros-L</td>
<td>7</td>
<td>41</td>
<td>39</td>
<td>2</td>
</tr>
<tr>
<td>Claros-LE</td>
<td>46</td>
<td>17.5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Pick memory in GB

<table>
<thead>
<tr>
<th></th>
<th>VLog</th>
<th>RDFox</th>
<th>X</th>
<th>TGs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Claros-L</td>
<td>3</td>
<td>5.4</td>
<td>6.4</td>
<td>6</td>
</tr>
<tr>
<td>Claros-LE</td>
<td>48</td>
<td>11.8</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

June 1, 2023 ESWC’2023
Reasoning at Scale: How Lineage Trigger Graphs

Aim

– Develop highly-scalable reasoning techniques that support uncertainty.
– Adopt well-established semantics.
Key Challenge: Complexity

Rules

\[ e(X, Y) \rightarrow p(X, Y) \]
\[ p(X, Z) \land p(Z, Y) \rightarrow p(X, Y) \]

Facts

\[ \rightarrow e(a, b) \]
\[ \rightarrow e(b, c) \]

Derivations

\[ \tau_5 \ p(a, c) \quad \tau_6 \ p(b, b) \quad \tau_7 \ p(a, b) \]
\[ \tau_1 \ p(a, b) \quad \tau_2 \ p(b, c) \quad \tau_3 \ p(a, c) \quad \tau_4 \ p(c, b) \]

\[ e(a, b) \quad e(b, c) \quad e(a, c) \quad e(c, b) \]
Prior Art: Key Limitations

– Relies on provenance semirings [9], i.e., associates a Boolean formula to each derivation.
– Super-polynomial size blowup in data complexity: any monotone formula to test connectivity in a graph with \( n \) nodes has size \( n^{\Omega(\log n)} \) (lower bound holds even for undirected graphs) [12].
– Requires Boolean checks at each reasoning step for termination.
– Runtime bottleneck.

## Probabilistic Reasoning via Provenance Semirings

<table>
<thead>
<tr>
<th>R</th>
<th>Derivation@R</th>
<th>Comparison</th>
<th>Formula@R</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$e(a, b)$</td>
<td>$\emptyset$</td>
<td>$e(a, b)$</td>
</tr>
<tr>
<td>2</td>
<td>$e(a, c) \land e(c, b)$</td>
<td>$e(a, c) \land e(c, b) \equiv e(a, b)$</td>
<td>$e(a, c) \land e(c, b) \lor e(a, b)$</td>
</tr>
</tbody>
</table>

\[\begin{array}{cccc}
\tau_5 & p(a, c) & \tau_6 & p(b, b) & \tau_7 & p(a, b) \\
\tau_1 & p(a, b) & \tau_2 & p(b, c) & \tau_3 & p(a, c) & \tau_4 & p(c, b) \\
\end{array}\]

\[\begin{array}{cccc}
e(a, b) & e(b, c) & e(a, c) & e(c, b) \\
\end{array}\]
Lineage Trigger Graphs

- Efficient maintenance of derivation history.
- Natural for TGs.
- Storing pointer offsets.
- Reduces termination checks for detecting cyclic derivations!
- No Boolean checks are required!
Lineage Trigger Graphs: (Adaptive) Provenance Circuits

- Extended the notion of provenance circuits [5] to allow a more space-efficient reasoning:
- Polynomial size representation.
Probabilistic Datalog Reasoning with Trigger Graphs

Figure: Time in seconds for goal-driven QA over sample queries from VQAR [11].
Conclusions++
Cool Research not Covered: Goal-driven QA over existential rules with equality (AAAI 2018)

Figure: Time in msec to answer the ChaseBench queries [2].

Cool Research not Covered: PRISM (AAAI 2023)

– **Objective**: mining rule patterns under \((\epsilon, \alpha)\)-guarantees:
  – \(\epsilon\) controls the uncertainty in the entity similarity measure;
  – \(\alpha\) controls the softness of the resulting rules.

– Runtime optimality for given \(\epsilon\).

– \(O(n \log n)\) vs. \(O(n^3)\) (in the size of the entities in the data) algorithm for clustering structurally-related data.

– PRISM outperforms SOTA by up to 6% in accuracy and up to 80% in runtime.

Keywords (instead of conclusions)

– Uncertainty– many proposals, what is the right semantics?
– Formal guarantees.
Thanks!

Contact info: efi.tsamoura@samsung.com.
References I


References II

Mark Chavira and Adnan Darwiche.
On probabilistic inference by weighted model counting.

Daniel Deutch, Tova Milo, Sudeepa Roy, and Val Tannen.
Circuits for datalog provenance.
In ICDT, pages 201–212, 2014.

Daan Fierens, Guy Van den Broeck, Joris Renkens, Dimitar Shterionov, Bernd Gutmann, Ingo Thon, Gerda Janssens, and Luc De Raedt.
Inference and learning in probabilistic logic programs using weighted boolean formulas.


References IV


References V

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Li Fei-Fei.
Visual genome: Connecting language and vision using crowdsourced dense image annotations.

Zijian Kuang and Xinran Tie.
Video understanding based on human action and group activity recognition.

N. Leone, G. Pfeifer, W. Faber, T. Eiter, G. Gottlob, S. Perri, and F. Scarcello.
The DLV system for knowledge representation and reasoning.


References VII


References VIII

Knot Pipatsrisawat and Adnan Darwiche.
New compilation languages based on structured decomposability.

Stephan Schulz.
System Description: E 1.8.
In *LPAR*, 2013.

Hao Tan and Mohit Bansal.
LXMERT: Learning cross-modality encoder representations from transformers.

Efthymia Tsamoura, Timothy Hospedales, and Loizos Michael.
Neural-symbolic integration: A compositional perspective.