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

Efi Tsamoura

Samsung AI, Cambridge, UK

Datalog Reasoning with Trigger Graphs

	1B	2B	4B	8B	17B
Runtime (s)	203	226	520	993	2272
Memory (GB)	23	34	49	98	174
#IDPs	1B	2B	5B	10B	20B

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

Probabilistic Datalog Reasoning with Lineage Trigger Graphs

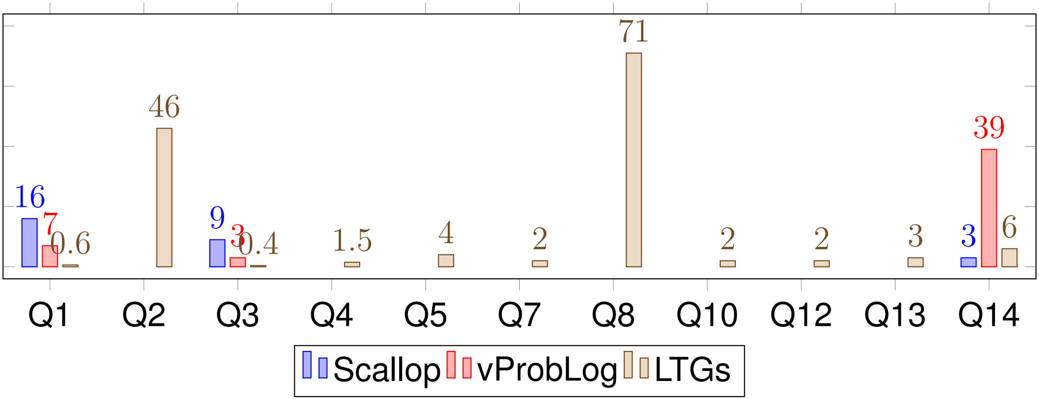
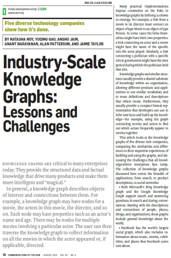


Figure: Time in seconds for goal-driven QA over probabilistic LUBM-100.

Reasoning (at Scale): Why

Why: Data Management



DOI:10.1145/3331166

Article development led by queue
queue.acm.org

Five diverse technology companies show how it's done.

BY NATASHA NOY, YUQING GAO, ANSHU JAIN, ANANT NARAYANAN, ALAN PATTERSON, AND JAMIE TAYLOR

Industry-Scale Knowledge Graphs: Lessons and Challenges

KNOWLEDGE GRAPHS ARE critical to many enterprises today: They provide the structured data and factual knowledge that drive many products and make them more intelligent and "magical."

In general, a knowledge graph describes objects of interest and connections between them. For example, a knowledge graph may have nodes for a movie, the actors in this movie, the director, and so on. Each node may have properties such as an actor's name and age. There may be nodes for multiple movies involving a particular actor. The user can then traverse the knowledge graph to collect information on all the movies in which the actor appeared or, if applicable, directed.

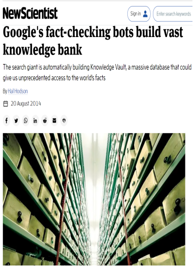
Many practical implementations impose constraints on the links in knowledge graphs by defining a *schema* or *ontology*. For example, a link from a movie to its director must connect an object of type Movie to an object of type Person. In some cases the links themselves might have their own properties: a link connecting an actor and a movie might have the name of the specific role the actor played. Similarly, a link connecting a politician with a specific role in government might have the time period during which the politician held that role.

Knowledge graphs and similar structures usually provide a shared substrate of knowledge within an organization, allowing different products and applications to use similar vocabulary and to reuse definitions and descriptions that others create. Furthermore, they usually provide a compact formal representation that developers can use to infer new facts and build up the knowledge—for example, using the graph connecting movies and actors to find out which actors frequently appear in movies together.

This article looks at the knowledge graphs of five diverse tech companies, comparing the similarities and differences in their respective experiences of building and using the graphs, and discussing the challenges that all knowledge-driven enterprises face today. The collection of knowledge graphs discussed here covers the breadth of applications, from search, to product descriptions, to social networks:

- Both Microsoft's Bing knowledge graph and the Google Knowledge Graph support search and answering questions in search and during conversations. Starting with the descriptions and connections of people, places, things, and organizations, these graphs include general knowledge about the world.
- Facebook has the world's largest social graph, which also includes information about music, movies, celebrities, and places that Facebook users care about.

36 COMMUNICATIONS OF THE ACM | AUGUST 2019 | VOL. 62 | NO. 8



NewScientist

Sign in

Enter search keywords

Google's fact-checking bots build vast knowledge bank

The search giant is automatically building Knowledge Vault, a massive database that could give us unprecedented access to the world's facts

By Hal Hodson

20 August 2014

- **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

- Build simpler models [11].
  - 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:
  - DeepProbLog [18]; Scallop [11].
  - NeuroLog [25]: abduction + WMC-based loss [4].

Efthymia Tsamoura and Loizos Michael **Neural-Symbolic Integration: a Compositional Perspective**. In AAAI, pages 5051-5060, 2021.

Can we Learn via Weak Supervision Coming from Logic? Yes

(*Work in progress*)

Theorem

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

$$\mathcal{R}^{01}(f) \leq O(\mathcal{R}^{01}_{\mathsf{P}}(f;\mathcal{G})^{1/M}) \quad \textit{as} \quad \mathcal{R}^{01}_{\mathsf{P}}(f;\mathcal{G}) \to 0$$

Furthermore, suppose $[\mathcal{F}]$ has a finite Natarajan dimension $d_{[\mathcal{F}]}$ and the function class $\{(\boldsymbol{y}, s) \mapsto 1\{\sigma'(\boldsymbol{y}) \neq s\} | \sigma' \in \mathcal{G}\}$ 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 $\widehat{\mathcal{R}}^{01}_{\mathsf{P}}(f;\sigma) = 0$ has a classification risk $\mathcal{R}^{01}(f) < \epsilon$, if

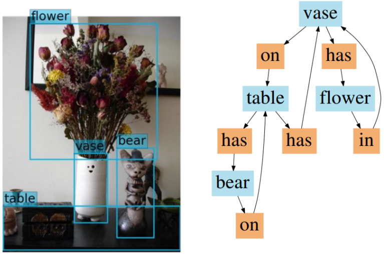
$$m_{\mathsf{P}} \geq C_4 \frac{c^{2M-2}}{r^M \epsilon^M} \left(\left((d_{[\mathcal{F}]} + d_{\mathcal{G}}) \log(6M(d_{[\mathcal{F}]} + d_{\mathcal{G}})) + d_{[\mathcal{F}]} \log c \right) \log \left(\frac{c^{2M-2}}{r^M \epsilon^M} \right) + \log \left(\frac{1}{\delta} \right) \right)$$

Believe in KRR

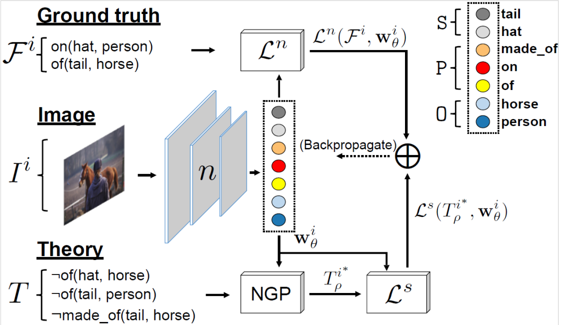
-My neurosymbolic research

Scene Graph Generation (AAAI 2023)

Task



Logic-Based Regularization



Davide Buffelli, and **Efthymia Tsamoura**. **Scalable Theory-Driven Regularization of Scene Graph Generation Models**. In AAAI, 2023.

Scene Graph Generation (AAAI 2023)

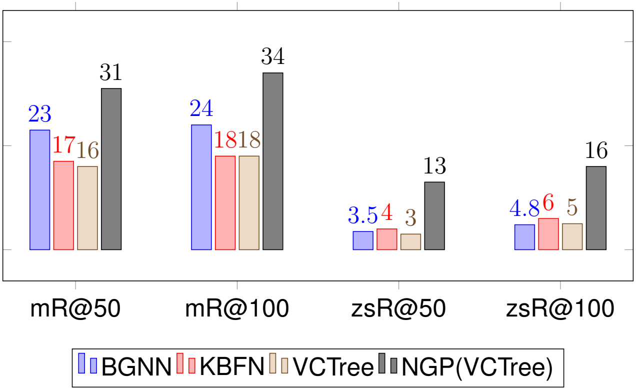


Figure: Comparison against BGNN [16], KBFN [10] and VCTree [22]. Benchmark: Visual Genome [13].

Scene Graph Generation (AAAI 2023)

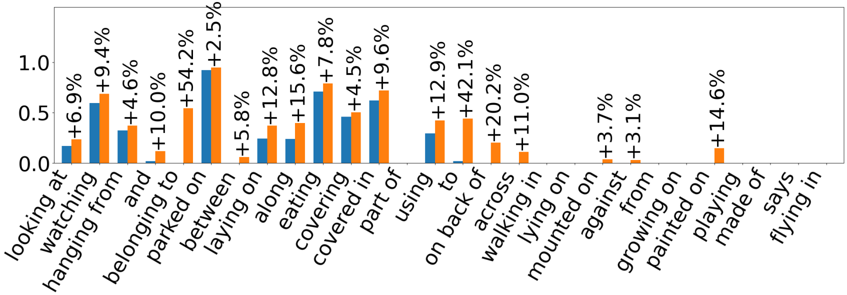
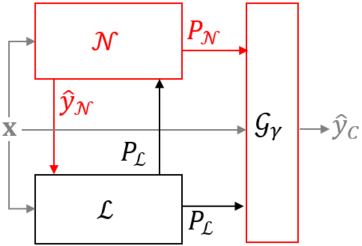


Figure: Recall of VCTree [22] on the 28 least frequent predicates: without NGP; with NGP. Benchmark: Visual Genome [13].

Knowledge Distillation into Deep Networks (ICML 2023)

Concordia

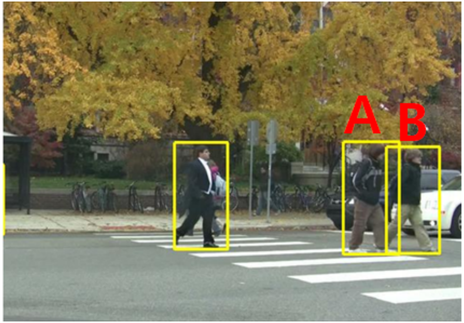


- First to support general first-order theories.
- Supports semi-/un-/supervised learning.

Operation	Equation
Inference	$\widehat{\mathbf{y}} = \arg\max_{\mathbf{y}} P_{\mathcal{N}}(\mathbf{Y} = \mathbf{y} \vert \mathbf{X} = \mathbf{x}, \boldsymbol{\theta})$
Training	$\widehat{\boldsymbol{\theta}}_{t+1} = \arg\min_{\boldsymbol{\theta}}(\ell(\widehat{\mathbf{y}}_{\mathcal{N}}, \mathbf{y}) + KL(P_{\mathcal{N}}, P_{\mathcal{L}}))$
	$\widehat{\boldsymbol{\lambda}}_{t+1} = \arg\max_{\boldsymbol{\lambda}} \prod_{(\mathbf{x}) \in \mathcal{D}} P_{\mathcal{L}}(\mathbf{X} = \mathbf{x}, \boldsymbol{\lambda}_t)$

Leon Jonathan Feldstein, Jurčius Modestas and **Efthymia Tsamoura**. **Parallel neurosymbolic integration with Concordia**. In ICML (*to appear*), 2023.

Video Activity Detection (ICML 2023)



$\mathsf{SEQ}(B_1, B_2) \wedge \mathsf{CLOSE}(B_1, B_2) \rightarrow \mathsf{SAME}(B_1, B_2)$

$\mathsf{DOING}(B_1, A) \wedge \mathsf{SAME}(B_1, B_2) \rightarrow \mathsf{DOING}(B_2, A)$

Accuracy over 5 runs

Model	Avg (%)	Max (%)	Min (%)
ACD+$\mathcal{L}$ [17]	86.00	-	-
MobileNet	90.07	91.36	89.61
IARG(MobileNet) [14]	90.18	92.39	87.55
Concordia(MobileNet, $\mathcal{L}$)	**90.73**	**93.19**	**89.54**
Inception	89.72	91.83	86.84
IARG(Inception) [14]	88.88	91.67	85.33
Concordia(Inception, $\mathcal{L}$)	**92.75**	**93.34**	**92.31**

Leon Jonathan Feldstein, Jurčius Modestas and **Efthymia Tsamoura**. **Parallel neurosymbolic integration with Concordia**. In ICML (*to appear*), 2023.

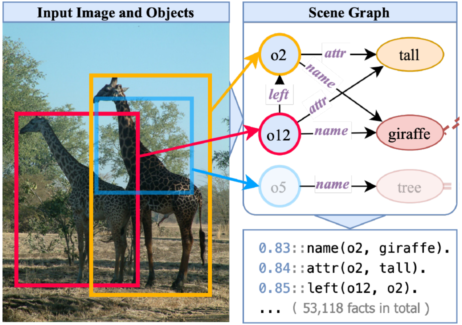
Entity Linking (ICML 2023)

Table: Results on entity linking.

Model	F_1	Acc (%)
BERT (sp)	0.88	**88.5**
Concordia(BERT) (sm)	**0.91**	**91.4**

Leon Jonathan Feldstein, Jurčius Modestas and **Efthymia Tsamoura**. **Parallel neurosymbolic integration with Concordia**. In ICML, 2023 (*to appear*).

Visual QA (SIGMOD 2023)



```
0.83::name(o2, giraffe).
0.84::attr(o2, tall).
0.85::left(o12, o2).
... ( 53,118 facts in total )
```

$$\mathsf{Q}(O) \leftarrow \mathsf{NAME}(herbivore, O)$$

$$\mathsf{NAME}(N, O) \wedge \mathsf{NAME}(N', O) \rightarrow \mathsf{ISA}(N', N)$$

$$\rightarrow \mathsf{ISA}(giraffe, herbivore)$$

$$\rightarrow \mathsf{ISA}(dear, herbivore)$$

Table: Recall@5 on VQAR [11].

Testset	LXMERT [24]	RVC [7]	TG-Guided VQA
C5	64.05%	74.62%	**87.01%**
C6	56.51%	72.04%	**85.45%**

Efthymia Tsamoura, Jaehun Lee, and Jacopo Urbani. **Probabilistic Reasoning as Scale: Trigger Graphs to the Rescue**. In SIGMOD, 2023 (*to appear*).

How this Reasoning Journey Started

Benchmarking the Chase (PODS 2017)

Benchmarking the Chase

Michael Benedikt [1] George Konstantinidis [1] Giansalvatore Mecca [2]
Boris Motik [1] Paolo Papotti [3] Donatello Santoro [2]
Efthymia Tsamoura [1]

[1] University of Oxford
Oxford, UK

[2] University of Basilicata
Potenza, Italy

[3] Arizona State University
Phoenix, AZ, USA

ABSTRACT

The chase is a family of algorithms used in a number of data management tasks, such as data exchange, answering queries under dependencies, query reformulation with constraints, and data cleaning. It is well established as a theoretical tool for understanding these tasks, and in addition a number of prototype systems have been developed. While individual chase-based systems and particular optimizations of the chase have been experimentally evaluated in the past, we provide the first comprehensive and publicly available benchmark—test infrastructure and a set of test scenarios—for evaluating chase implementations across a wide range of assumptions about the dependencies and the data. We used our benchmark to compare chase-based systems on data exchange and query answering tasks with one another, as well as with systems that can solve similar tasks developed in closely related communities. Our evaluation provided us with a number of new insights concerning the factors that impact the performance of chase implementations.

1. INTRODUCTION

The chase [25] is a long-standing technique developed by the database community for reasoning with constraints, also known as *dependencies*, expressed as universal implications possibly containing existential quantification in the conclusion. When applied to a set of dependencies and a set of facts, the chase extends the facts in a forward-chaining manner to satisfy the dependencies.

The chase has been intensively studied as a theoretical tool, but over the last decade practical aspects, such as developing optimizations of the chase algorithms and building chase-based systems for various tasks, have also been considered. Although algorithms for chasing source-to-target dependencies have shown promise in practice, with target dependencies scalability has been achieved only in quite restricted cases. The performance of chase implementations on large sets of complex dependencies and large datasets remains unknown. This suggests that it is time to evaluate the extent to which computing the chase is practically feasible.

The chase is closely related to and can be seen as a special case of theorem proving calculi such as tableaux and resolution, and it can also be seen as a generalization of standard query evaluation in databases. But while the theorem proving and the database communities have a long history of creating benchmarks and detailed evaluation methodologies (e.g., SMTLib [37] and TPTP [40] in the former, the TPC family [39] in the latter), there is little corresponding infrastructure to support experimental validation of techniques such as the chase that combine reasoning and data management.

This paper aims to take a major step in changing this situation. We present a new benchmark for chase systems covering a wide range of scenarios. Since the systems in the literature support different kinds of dependencies, we have developed dependency sets with different structural properties, and datasets of varying sizes.

We also define example tasks for two main applications of the chase: (i) data exchange, which involves materializing an instance of a target schema satisfying a given set of dependencies with respect to an instance of a source schema; and (ii) computing certain answers to conjunctive queries over databases with dependencies.

We then analyze a variety of publicly available systems on our benchmark in order to answer the following questions:

- How do existing chase-related systems fare in absolute terms on these tasks? That is, to what extent can they be considered as proof that the chase-based approaches to solving these tasks are practically feasible?
- What algorithmic and architectural choices are most critical for the performance of chase-related systems?
- Are there other approaches or other kinds of systems that can perform these same tasks and, if so, how do they compare to tools that use the chase?

In an attempt to answer these questions, we considered a number of systems that implement the chase as a component, including systems motivated from data exchange, data cleaning, query reformulation, and query answering.

We mentioned above that many communities have looked at techniques similar to the chase, and at problems similar to data exchange and query answering. To better understand the connection with the related communities, we also applied our benchmark to systems that are not specifically "branded" as chase systems, but that can nonetheless perform some of the tasks that the chase addresses. In particular, we looked at Datalog engines that support function symbols as they can solve both data exchange and query answering problems, as well as a leading theorem prover that can solve various query answering problems.

Organization. In the rest of this paper, we first present some background about the chase (Sections 2 and 3). Next, we describe our test systems (Section 4), and discuss our testing infrastructure and test scenarios (Section 5). Then, we present the system comparison results (Section 6), followed by a discussion of the insights gained (Section 7) and the future challenges that emerged from our study (Section 8). Finally, we close with a discussion of the re-

PODS'17, May 14-19, 2017, Chicago, Illinois, USA
© 2017 ACM. ISBN 978-1-4503-4198-1/17/05. . . $15.00
DOI: http://dx.doi.org/10.1145/3034786.3034796

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

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

Michael Benedikt, George Konstantinidis, Giansalvatore Mecca, Boris Motik, Paolo Papotti, Donatello Santoro, and **Efthymia Tsamoura**. **Benchmarking the Chase**. In PODS, pages 37–52, 2017.

Benchmarking the Chase (PODS 2017)

Benchmarking the Chase

Michael Benedikt[1] George Konstantinidis[1] Giansalvatore Mecca[2]
Boris Motik[1] Paolo Papotti[3] Donatello Santoro[2]
Efthymia Tsamoura[1]

[1] University of Oxford
Oxford, UK

[2] University of Basilicata
Potenza, Italy

[3] Arizona State University
Phoenix, AZ, USA

ABSTRACT

The chase is a family of algorithms used in a number of data management tasks, such as data exchange, answering queries under dependencies, query reformulation with constraints, and data cleaning. It is well established as a theoretical tool for understanding these tasks, and in addition a number of prototype systems have been developed. While individual chase-based systems and particular optimizations of the chase have been experimentally evaluated in the past, we provide the first comprehensive and publicly available benchmark—test infrastructure and a set of test scenarios—for evaluating chase implementations across a wide range of assumptions about the dependencies and the data. We used our benchmark to compare chase-based systems on data exchange and query answering tasks with one another, as well as with systems that can solve similar tasks developed in closely related communities. Our evaluation provided us with a number of new insights concerning the factors that impact the performance of chase implementations.

1. INTRODUCTION

The chase [25] is a long-standing technique developed by the database community for reasoning with constraints, also known as *dependencies*, expressed as universal implications possibly containing existential quantification in the conclusion. When applied to a set of dependencies and a set of facts, the chase extends the facts in a forward-chaining manner to satisfy the dependencies.

The chase has been intensively studied as a theoretical tool, but over the last decade practical aspects, such as developing optimizations of the chase algorithms and building chase-based systems for various tasks, have also been considered. Although algorithms for chasing source-to-target dependencies have shown promise in practice, with target dependencies scalability has been achieved only in quite restricted cases. The performance of chase implementations on large sets of complex dependencies and large datasets remains unknown. This suggests that it is time to evaluate the extent to which computing the chase is practically feasible.

The chase is closely related to and can be seen as a special case of theorem proving calculi such as tableaux and resolution, and it can also be seen as a generalization of standard query evaluation in databases. But while the theorem proving and the database communities have a long history of creating benchmarks and detailed evaluation methodologies (e.g., SMTLib [37] and TPTP [40] in the former, the TPC family [39] in the latter), there is little corresponding infrastructure to support experimental validation of techniques such as the chase that combine reasoning and data management.

This paper aims to take a major step in changing this situation. We present a new benchmark for chase systems covering a wide range of scenarios. Since the systems in the literature support different kinds of dependencies, we have developed dependency sets with different structural properties, and datasets of varying sizes.

We also define example tasks for two main applications of the chase: (i) data exchange, which involves materializing an instance of a target schema satisfying a given set of dependencies with respect to an instance of a source schema; and (ii) computing certain answers to conjunctive queries over databases with dependencies.

We then analyze a variety of publicly available systems on our benchmark in order to answer the following questions:

- How do existing chase-related systems fare in absolute terms on these tasks? That is, to what extent can they be considered as proof that the chase-based approaches to solving these tasks are practically feasible?
- What algorithmic and architectural choices are most critical for the performance of chase-related systems?
- Are there other approaches or other kinds of systems that can perform these same tasks and, if so, how do they compare to tools that use the chase?

In an attempt to answer these questions, we considered a number of systems that implement the chase as a component, including systems motivated from data exchange, data cleaning, query reformulation, and query answering.

We mentioned above that many communities have looked at techniques similar to the chase, and at problems similar to data exchange and query answering. To better understand the connection with the related communities, we also applied our benchmark to systems that are not specifically "branded" as chase systems, but that can nonetheless perform some of the tasks that the chase addresses. In particular, we looked at Datalog engines that support function symbols as they can solve both data exchange and query answering problems, as well as a leading theorem prover that can solve various query answering problems.

Organization. In the rest of this paper, we first present some background about the chase (Sections 2 and 3). Next, we describe our test systems (Section 4), and discuss our testing infrastructure and test scenarios (Section 5). Then, we present the system comparison results (Section 6), followed by a discussion of the insights gained (Section 7) and the future challenges that emerged from our study (Section 8). Finally, we close with a discussion of the re-

PODS'17, May 14-19, 2017, Chicago, Illinois, USA
© 2017 ACM. ISBN 978-1-4503-4198-1/17/05. . . $15.00
DOI: http://dx.doi.org/10.1145/3034786.3034796

- **Paper takeaways**
  - Equality is challening.
  - 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.

Michael Benedikt, George Konstantinidis, Giansalvatore Mecca, Boris Motik, Paolo Papotti, Donatello Santoro, and **Efthymia Tsamoura**. **Benchmarking the Chase**. In PODS, pages 37–52, 2017.

How this Journey Started (cont'): ProbLog

Under consideration for publication in Theory and Practice of Logic Programming 1

Inference and Learning in Probabilistic Logic Programs using Weighted Boolean Formulas

DAAN FIERENS, GUY VAN DEN BROECK, JORIS RENKENS,
DIMITAR SHTERIONOV, BERND GUTMANN, INGO THON,
GERDA JANSSENS, LUC DE RAEDT
Department of Computer Science
KU Leuven
Celestijnenlaan 200A, 3001 Heverlee, Belgium
(*e-mail:* `FirstName.LastName@cs.kuleuven.be`)

submitted 26 June 2012; revised 04 January 2013; accepted 07 March 2013

Abstract

Probabilistic logic programs are logic programs in which some of the facts are annotated with probabilities. This paper investigates how classical inference and learning tasks known from the graphical model community can be tackled for probabilistic logic programs. Several such tasks such as computing the marginals given evidence and learning from (partial) interpretations have not really been addressed for probabilistic logic programs before.

The first contribution of this paper is a suite of efficient algorithms for various inference tasks. It is based on a conversion of the program and the queries and evidence to a weighted Boolean formula. This allows us to reduce the inference tasks to well-studied tasks such as weighted model counting, which can be solved using state-of-the-art methods known from the graphical model and knowledge compilation literature. The second contribution is an algorithm for parameter estimation in the learning from interpretations setting. The algorithm employs Expectation Maximization, and is built on top of the developed inference algorithms.

The proposed approach is experimentally evaluated. The results show that the inference algorithms improve upon the state-of-the-art in probabilistic logic programming and that it is indeed possible to learn the parameters of a probabilistic logic program from interpretations.

KEYWORDS: Probabilistic logic programming, Probabilistic inference, Parameter learning

1 Introduction

There is a lot of interest in combining probability and logic for dealing with complex relational domains. This interest has resulted in the fields of *Probabilistic Logic Programming (PLP)* (De Raedt et al. 2008) and *Statistical Relational Learning (SRL)* (Getoor and Taskar 2007). While the two fields essentially study the same problem, there are differences in emphasis. SRL techniques have focussed on the extension of probabilistic graphical models like Markov or Bayesian networks with

- **Why ProbLog [6]**
  - 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.

Efthymia Tsamoura, Victor Gutierrez-Basulto, and Angelika Kimmig. **Beyond the Grounding Bottleneck: Datalog Techniques for Inference in Probabilistic Logic Programs**. In AAAI, pages 10284-10291, 2020.

Reasoning at Scale: How -Trigger Graphs

Efthymia Tsamoura, David Carral, Enrico Malizia, and Jacopo Urbani. **Materializing Knowledge Bases via Trigger Graphs**. In VLDB, pages 943-951, 2021.

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

$$r(X,Y) \rightarrow R(X,Y) \tag{r_1}$$

$$R(X,Y) \rightarrow T(Y,X,Y) \tag{r_2}$$

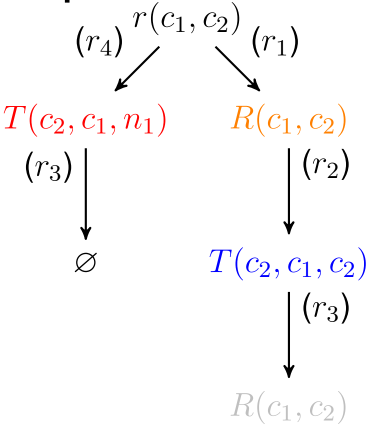
$$T(Y,X,Y) \rightarrow R(X,Y) \tag{r_3}$$

$$r(X,Y) \rightarrow \exists Z.T(Y,X,Z) \tag{r_4}$$

Facts

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

Bottom-Up evaluation



$r(c_1, c_2)$

(r_4)

(r_1)

$T(c_2, c_1, n_1)$

$R(c_1, c_2)$

(r_3)

(r_2)

$\varnothing$

$T(c_2, c_1, c_2)$

(r_3)

$R(c_1, c_2)$

How: Trigger Graphs

Rules

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

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

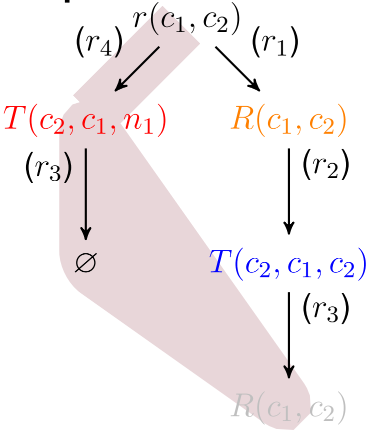
$$T(Y,X,Y) \rightarrow R(X,Y) \qquad (r_3)$$

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

Facts

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

Bottom-Up evaluation



$r(c_1,c_2)$

(r_4)

(r_1)

$T(c_2,c_1,n_1)$

$R(c_1,c_2)$

(r_3)

(r_2)

$\varnothing$

$T(c_2,c_1,c_2)$

(r_3)

$R(c_1,c_2)$

How: Trigger Graphs

Rules

$$r(X,Y) \to R(X,Y) \qquad (r_1)$$

$$R(X,Y) \to T(Y,X,Y) \qquad (r_2)$$

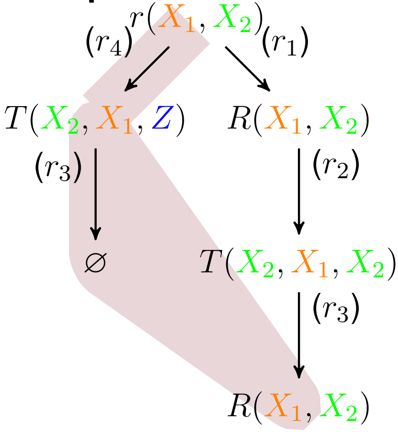
$$T(Y,X,Y) \to R(X,Y) \qquad (r_3)$$

$$r(X,Y) \to \exists Z.T(Y,X,Z) \qquad (r_4)$$

Facts

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

Bottom-Up evaluation



How: Trigger Graphs

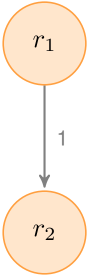
Rules

$$r(X,Y) \to R(X,Y) \qquad (r_1)$$
$$R(X,Y) \to T(Y,X,Y) \qquad (r_2)$$
$$T(Y,X,Y) \to R(X,Y) \qquad (r_3)$$
$$r(X,Y) \to \exists Z. T(Y,X,Z) \qquad (r_4)$$

Facts

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

Trigger graph



Trigger graph-based reasoning

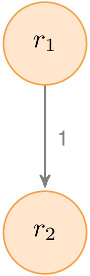
TGs delineate the rule executions

- 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



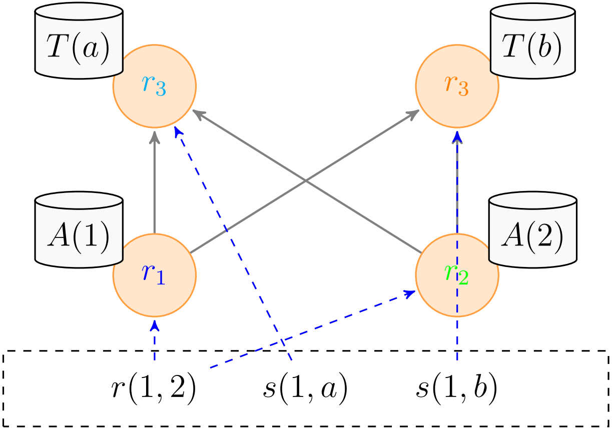
Trigger graph-based reasoning

Rules

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

$$r(X,Y) \rightarrow A(Y) \quad (r_2)$$

$$A(X) \wedge s(X,Z) \rightarrow T(Z) \quad (r_3)$$



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 redundanct 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

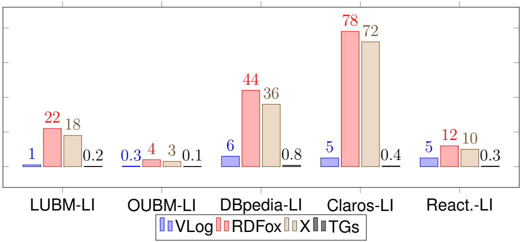
Let P be a linear program that admits a finite universal model.

Theorem (Complexity)

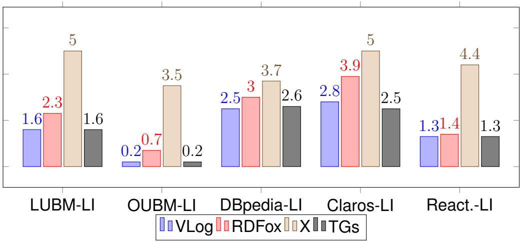
Computing a TG for P is double exponential in P. If the arity of the predicates in P is bounded, the computation time is (single) exponential.

Reasoning over Linear Rules

Total materialization times in s



Pick memory in GB



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].

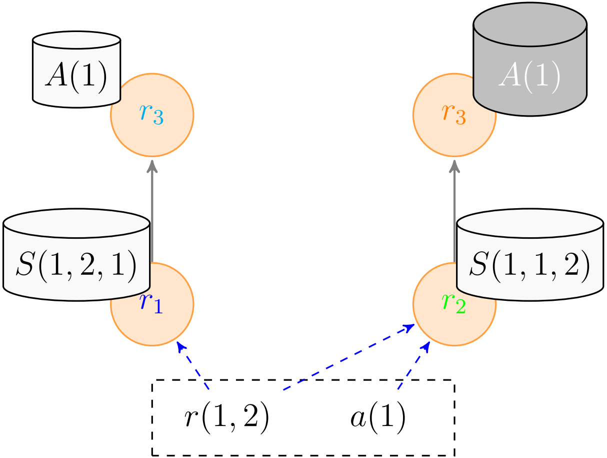
Trigger Graphs for Datalog Rules: Example

Rules

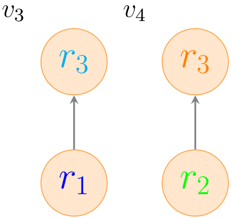
$$r(X,Y) \rightarrow S(X,Y,X) \quad (1)$$

$$a(X) \wedge r(X,Y) \rightarrow S(X,X,Y) \quad (2)$$

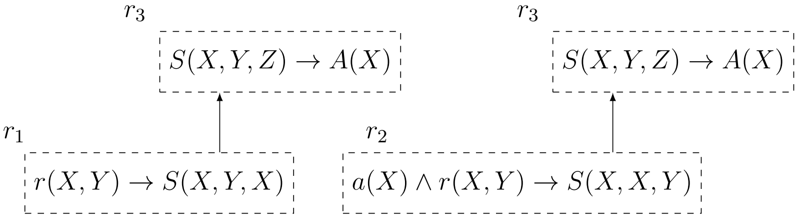
$$S(X,Y,Z) \rightarrow A(X) \quad (3)$$



Trigger Graphs for Datalog Rules: Example



Trigger Graph



r_3: $S(X, Y, Z) \rightarrow A(X)$

r_1: $r(X, Y) \rightarrow S(X, Y, X)$

r_3: $S(X, Y, Z) \rightarrow A(X)$

r_2: $a(X) \wedge r(X, Y) \rightarrow S(X, X, Y)$

$$\boxed{Q(X) = \exists Y . r(X, Y)}$$

Query for v_3

$$\boxed{Q'(X) = \exists Y . a(X) \wedge r(X, Y)}$$

Query for v_4

Trigger Graphs for Datalog Rules: Results

Let P be a Datalog program.

Theorem (Soundness)

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

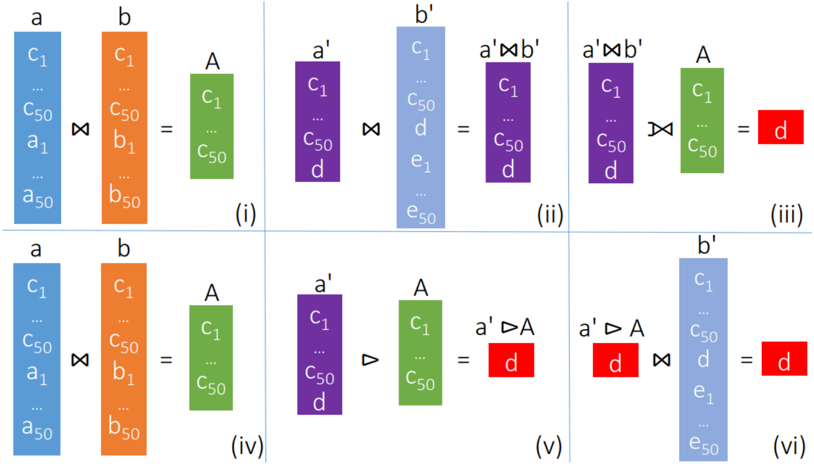
Theorem (Minimality)

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

Theorem (Complexity)

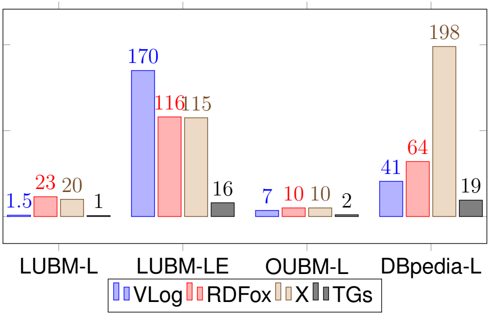
Deciding whether G is a TG of minimum size for P is co-NP-complete.

More: TG-Aware Rule Execution Strategy

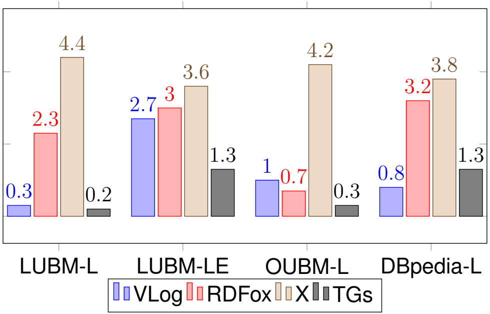


Datalog Reasoning with Trigger Graphs

Materialization times in s

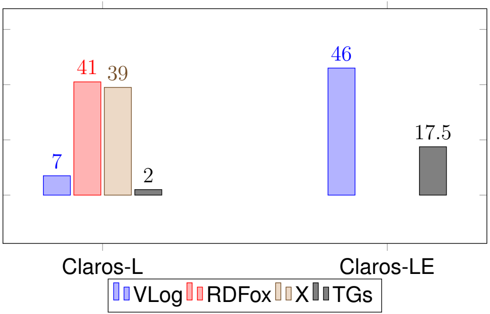


Pick memory in GB

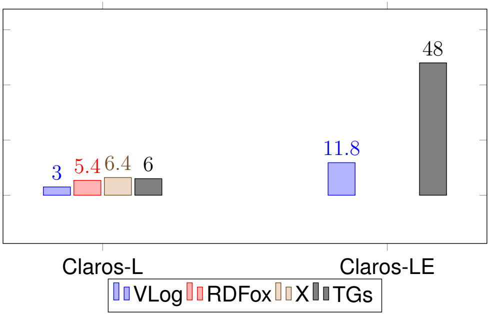


Datalog Reasoning with Trigger Graphs

Materialization times in minutes



Pick memory in GB



Reasoning at Scale: How -Lineage Trigger Graphs

Efthymia Tsamoura, Jaehun Lee, and Jacopo Urbani. **Probabilistic Reasoning as Scale: Trigger Graphs to the Rescue**. In SIGMOD, 2023 (*to appear*).

Aim

- Develop highly-scalable reasoning techniques that support uncertainty.
- Adopt well-established semantics.

Key Challenge: Complexity

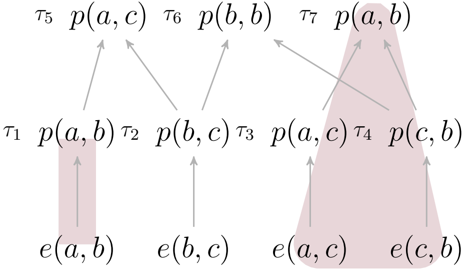
Rules

$$e(X, Y) \to p(X, Y)$$
$$p(X, Z) \land p(Z, Y) \to p(X, Y)$$

Facts

$$\to e(a, b) \qquad \to e(a, c)$$
$$\to e(b, c) \qquad \to e(c, b)$$

Derivations



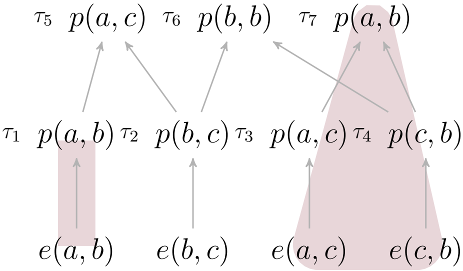
Prior Art: Key Limitations

- 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.

Efthymia Tsamoura, Victor Gutierrez-Basulto, and Angelika Kimmig. **Beyond the Grounding Bottleneck: Datalog Techniques for Inference in Probabilistic Logic Programs**. In AAAI, pages 10284-10291, 2020.

Probabilistic Reasoning via Provenance Semirings

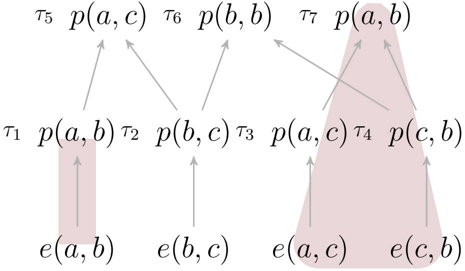
R	Derivation@R	Comparison	Formula@R
1	$e(a,b)$	$\varnothing$	$e(a,b)$
2	$e(a,c) \wedge e(c,b)$	$e(a,c) \wedge e(c,b) \overset{?}{\equiv} e(a,b)$	$e(a,c) \wedge e(c,b) \vee e(a,b)$



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!

Derivations



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

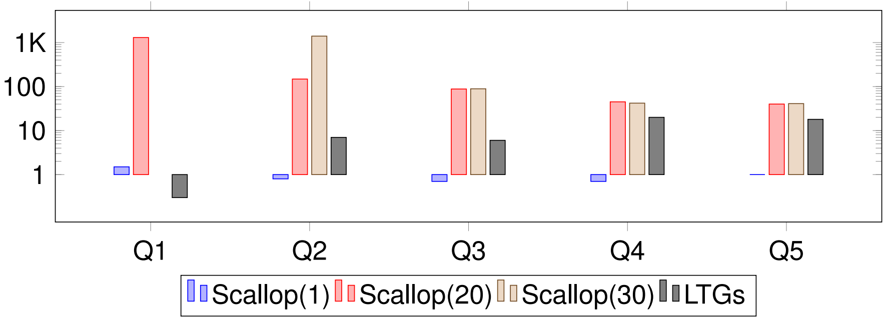


Figure: Time in seconds for goal-driven QA over sample queries from VQAR [11].

Conclusions++

Cool Research not Covered: Goal-driven QA over existential rules with equality (AAAI 2018)

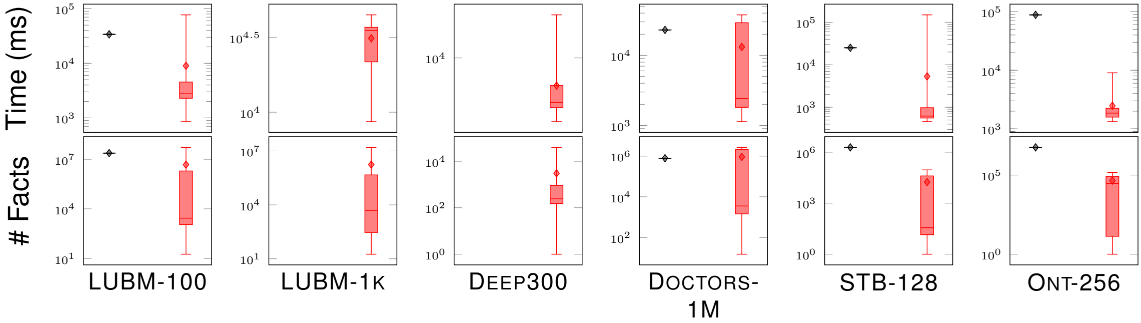


Figure: Time in msec to answer the ChaseBench queries [2].

Michael Benedikt, Boris Motik, and **Efthymia Tsamoura**. **Goal-Driven Query Answering over Existential Rules with Equality**. In AAAI, pages 1761–1770, 2018.

Cool Research not Covered: PRISM (AAAI 2023)

- **Objective**: mining rule patterns under (ϵ,α)-guarantees:
  - ϵ controls the uncertainty in the entity similarity measure;
  - α controls the softness of the resulting rules.
- Runtime optimality for given ϵ.
- $O(n \log n)$ 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.

Leon Jonathan Feldstein, Dominic Phillips and **Efthymia Tsamoura**. **Principled and Efficient Motif Finding for Structure Learning of Lifted Graphical Models**. In AAAI, 2023.

Keywords (instead of conclusions)

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

Thanks!

Contact info: `efi.tsamoura@samsung.com`.

References I

J.-F. Baget, M. Leclère, M.-L. Mugnier, S. Rocher, and C. Sipieter.
Graal: A toolkit for query answering with existential rules.
In *RuleML*, 2015.

Michael Benedikt, George Konstantinidis, Giansalvatore Mecca, Boris Motik, Paolo Papotti, Donatello Santoro, and Efthymia Tsamoura.
Benchmarking the Chase.
In *PODS*, pages 37–52, 2017.

Ashok K. Chandra and Philip M. Merlin.
Optimal implementation of conjunctive queries in relational data bases.
In *STOC*, pages 77–90, 1977.

References II

Mark Chavira and Adnan Darwiche.
On probabilistic inference by weighted model counting.
Artif. Intell., 172(6-7):772–799, 2008.

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.
Theory and Practice of Logic Programming (TPLP), 15(3):358–401, 2015.

References III

- Difei Gao, Ruiping Wang, Shiguang Shan, and Xilin Chen.
  From two graphs to N questions: A VQA dataset for compositional reasoning on vision and commonsense.
  CoRR, abs/1908.02962, 2019.

- Yuxia Geng, Jiaoyan Chen, Wen Zhang, Yajing Xu, Zhuo Chen, Jeff Z. Pan, Yufeng Huang, Feiyu Xiong, and Huajun Chen.
  Disentangled ontology embedding for zero-shot learning.
  In Aidong Zhang and Huzefa Rangwala, editors, *KDD*, pages 443–453. ACM, 2022.

- Todd J. Green, Grigoris Karvounarakis, and Val Tannen.
  Provenance semirings.
  In *PODS*, page 31–40, 2007.

References IV

Jiuxiang Gu, Handong Zhao, Zhe Lin, Sheng Li, Jianfei Cai, and Mingyang Ling.
Scene graph generation with external knowledge and image reconstruction.
In *CVPR*, pages 1969–1978, 2019.

Jiani Huang, Ziyang Li, Binghong Chen, Karan Samel, Mayur Naik, Le Song, and Xujie Si.
Scallop: From probabilistic deductive databases to scalable differentiable reasoning.
In *NeurIPS*, pages 25134–25145, 2021.

Mauricio Karchmer and Avi Wigderson.
Monotone circuits for connectivity require super-logarithmic depth.
In *STOC*, page 539–550, 1988.

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.
Int. J. Comput. Vis., 123(1):32–73, 2017.

Zijian Kuang and Xinran Tie.
Video understanding based on human action and group activity recognition.
CoRR, abs/2010.12968, 2020.

N. Leone, G. Pfeifer, W. Faber, T. Eiter, G. Gottlob, S. Perri, and F. Scarcello.
The DLV system for knowledge representation and reasoning.
TOCL, 7(3):499–562, 2006.

References VI

Rongjie Li, Songyang Zhang, Bo Wan, and Xuming He.
Bipartite graph network with adaptive message passing for unbiased scene graph generation.
In *CVPR*, pages 11109–11119, 2021.

B. London, S. Khamis, S. H. Bach, B. Huang, L. Getoor, and L. Davis.
Collective activity detection using hinge-loss markov random fields.
In *CVPR Workshops*, pages 566–571, 2013.

Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt.
Deepproblog: Neural probabilistic logic programming.
In *NeurIPS*, pages 3749–3759, 2018.

References VII

M. Meier.
The backchase revisited.
VLDB J., 23(3):495–516, 2014.

Yavor Nenov, Robert Piro, Boris Motik, Ian Horrocks, Zhe Wu, and Jay Banerjee.
RDFox: A Highly-Scalable RDF Store.
In *ISWC*, pages 3–20, 2015.

Natasha Noy, Yuqing Gao, Anshu Jain, Anant Narayanan, Alan Patterson, and Jamie Taylor.
Industry-scale Knowledge Graphs: Lessons and Challenges.
Commun. ACM, 62(8):36–43, 2019.

References VIII

Knot Pipatsrisawat and Adnan Darwiche.
New compilation languages based on structured decomposability.
In *AAAI*, page 517–522, 2008.

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

Hao Tan and Mohit Bansal.
LXMERT: Learning cross-modality encoder representations from transformers.
In *EMNLP*, pages 5100–5111, 2019.

Efthymia Tsamoura, Timothy Hospedales, and Loizos Michael.
Neural-symbolic integration: A compositional perspective.
In *AAAI*, pages 5051–5060, 2021.

References IX