# Scalable Integration of Uncertainty Reasoning and Semantic Web Technologies

Inauguraldissertation
zur Erlangung des akademischen Grades
eines Doktors der Naturwissenschaften
der Universität Mannheim

vorgelegt von

Jörg Schönfisch
aus Ludwigshafen am Rhein

Mannheim, 2018

|  |  |
|---|---|
| Dekan: | Dr. Bernd Lübcke, Universität Mannheim |
| Referent: | Prof. Dr. Heiner Stuckenschmidt, Universität Mannheim |
| Korreferent: | Prof. Dr. Kristian Kersting, Technische Universität Darmstadt |
|  |  |
| Tag der mündlichen Prüfung: | 25. Oktober 2018 |

# Abstract

In recent years formal logical standards for knowledge representation to model real world knowledge and domains and make them accessible for computers gained a lot of traction. They provide an expressive logical framework for modeling, consistency checking, reasoning, and query answering, and have proven to be versatile methods to capture knowledge of various fields. Those formalisms and methods focus on specifying knowledge as precisely as possible.

At the same time, many applications in particular on the Semantic Web have to deal with uncertainty in their data; and handling uncertain knowledge is crucial in many real-world domains. However, regular logic is unable to capture the real-world properly due to its inherent complexity and uncertainty, all the while handling uncertain or incomplete information is getting more and more important in applications like expert system, data integration or information extraction.

The overall objective of this dissertation is to identify scenarios and datasets where methods that incorporate their inherent uncertainty improve results, and investigate approaches and tools that are suitable for the respective task. In summary, this work is set out to tackle the following objectives:

1. debugging uncertain knowledge bases in order to generate consistent knowledge graphs to make them accessible for logical reasoning,
2. combining probabilistic query answering and logical reasoning which in turn uses these consistent knowledge graphs to answer user queries, and
3. employing the aforementioned techniques to the problem of risk management in IT infrastructures, as a concrete real-world application.

We show that in all those scenarios, users can benefit from incorporating uncertainty in the knowledge base. Furthermore, we conduct experiments that demonstrate the real-world scalability of the demonstrated approaches. Overall, we argue that integrating uncertainty and logical reasoning, despite being theoretically intractable, is feasible in real-world application and warrants further research.

# Zusammenfassung

In den letzten Jahren haben logische Formalism für die Wissensrepräsentation, die es ermöglichen Wissen und Domänen zu modellieren und sie für Computer zugänglich zu machen, an großer Bedeutung gewonnen. Sie bieten einen aussagekräftigen logischen Rahmen für Modellierung, Konsistenzprüfung, Argumentation und zur Beantwortung von Abfragen, und haben sich als vielseitige Methoden zur Erfassung von Wissen in verschiedensten Bereichen erwiesen. Diese Formalismen und Methoden konzentrieren sich darauf, Wissen so genau wie möglich zu spezifizieren.

Gleichzeitig müssen viele Anwendungen, insbesondere im Semantic Web, mit Unsicherheiten in ihren Daten umgehen; die Berücksichtigung unsicherem Wissen ist in vielen Bereichen der realen Welt von entscheidender Bedeutung. Formelle Logik ist jedoch nicht in der Lage die reale Welt aufgrund ihrer Komplexität und Unsicherheit korrekt zu erfassen. Allerdings wird der Umgang mit unsicheren oder unvollständigen Informationen in Anwendungen wie Expertensystem, Datenintegration oder Informationsextraktion immer wichtiger.

Das übergeordnete Ziel dieser Dissertation ist es, Szenarien und Datensätze zu identifizieren, bei denen Methoden, die ihre inhärente Unsicherheit berücksichtigen, Ergebnisse verbessern und Ansätze und Werkzeuge zu untersuchen, die für die jeweilige Aufgabe geeignet sind. Zusammenfassend ist diese Arbeit auf die folgenden Ziele ausgerichtet:

1. Debuggen unsicherer Wissensbasen, um konsistente Wissensgraphen zu erzeugen, die dadurch für logische Inferenz zugänglich werden,
2. die Kombination von logischer Inferenz und dem probabilistischen Beantworten von Anfragen, was wiederum konsistente Wissenbasen vorraussetzt, und
3. die Anwendung der oben genannten Techniken auf das Problem des Risikomanagements in IT-Infrastrukturen als eine konkrete reale Anwendung.

Wir zeigen, dass Nutzer in all diesen Szenarien davon profitieren können, wenn Unsicherheit in der Wissensbasis benhandelt wird. Darüber hinaus zeigen wir anhand von Experimenten, dass die angeführten Ansätze in realistischen Anwendungsszenarien skalierbar sind. Insgesamt argumentieren wir, dass die Integration von Unsicherheit und logischer Inferenz, obwohl theoretisch schlecht skalierbar, in realistischen Szenarien durchaus anwendbar ist.

# Contents

# List of Figures

# List of Tables

# List of Theorems

# Part I.

# Introduction

# 1 Introduction

## 1.1. Motivation

In recent years several standards for knowledge representation, e.g. the Resource Description Framework (RDF)[1] (Schreiber and Raimond, 2014) or the Web Ontology Language (OWL)[2] (Parsia et al., 2012) by the World Wide Web Consortium (W3C) gained a lot of traction. There are also some prominent de facto standards and formalisms like schema.org Guha et al. (2016) or Linked Open Data. In general, the purpose of these Semantic Web technologies is to model real world knowledge and domains and make them accessible for computers. They provide an expressive framework for modeling, consistency checking, reasoning, and query answering, and have proven to be versatile methods to capture knowledge of various fields. Those formalisms and methods focus on specifying knowledge as precisely as possible. OntoClean (Guarino and Welty, 2009), for example, categorizes expressions in an ontology into rigid and non-rigid ones and discourages the usage of non-rigid knowledge to avoid false reasoning results.

At the same time, many applications in particular on the Semantic Web have to deal with uncertainty in their data; and handling uncertain knowledge is crucial in many real-world domains. However, Russell et al. (2010, p. 481) note that regular logic is unable to capture the real-world properly due to its inherent complexity and uncertainty, and illustrate this by the following example:

**Example 1.1** (Logic and Uncertainty)
*Consider a simple logical rule which says that a toothache is implying that a person has a cavity:*

$$Toothache \rightarrow Cavity \tag{1.1}$$

*However, this rule is too simple, as there are other reasons for toothache, like gum problems:*

$$Toothache \rightarrow Cavity \lor GumProblems \lor \ldots \tag{1.2}$$

*Obviously, making this rule complete requires us to add an almost infinite number of causes for toothache, which is infeasible to do. We can turn the rule around, stating a*

[1]http://www.w3.org/standards/techs/rdf
[2]http://www.w3.org/standards/techs/owl

*causal implication:*

$$Cavity \rightarrow Toothache \tag{1.3}$$

*but again, this is not sufficient as not all cavities cause a toothache, and we have to add multiple preconditions as to when the ache really occurs. For both rules, the only solution to making them valid in the real-world is to make them logically exhaustive and add all causes and preconditions for toothache.* △

Russell et al. (2010) name three main reasons why logic fails like this in the real world:

- **Laziness:** It takes too much effort to exhaustively model such rules, and it makes those rules hard to use.
- **Theoretical ignorance:** For many domains, there exists no complete theory, e.g. not every cause of an observation is known.
- **Practical ignorance:** Even if there is a complete theory for a domain, all the relevant information for a concrete instantiation of the rule might not be available, for example we are still waiting for results of some medical test.

The Semantic Web – with description logic as its formal underpinning – is also hindered by these limitations. The impact for Linked Data or ontologies can be so severe that IBM decided not to use structured knowledge bases for their famous Jeopardy-winning Watson system, as it would only have been able to answer about two percent of the posed questions (Kalyanpur et al., 2012). Instead, they developed a system that allows for imprecise or uncertain knowledge and achieves a better overall performance. The handling of uncertain or incomplete information is getting more and more important, not only in the domain of expert systems or query-answering systems like Watson, but also in other real world applications like data integration and information extraction. Some examples for sources of uncertainty from those areas are:

- Data entered manually by some person like in Wikipedia[3] or Wikidata[4], a bug report from a user, or a filing from a worker. A user can either knowingly (like when changing Wikipedia articles about a person to put them in a more favorable light) or unknowingly (e.g. when a user is unaware about the correct version of a used product) enter wrong information.
- Data obtained through open information extraction (OIE), like the knowledge bases created by NELL[5] (Carlson et al., 2010), YAGO[6] (Hoffart et al., 2013), Microsoft's Concept Graph[7] (Wu et al., 2012), or Google's Knowledge Vault[8] (Dong

[3]https://www.wikipedia.org/

[4]https://www.wikidata.org/

[5]http://rtw.ml.cmu.edu/rtw/

[6]http://www.yago-knowledge.org/

[7]https://concept.research.microsoft.com/Home/Introduction

[8]https://www.google.com/intl/bn/insidesearch/features/search/knowledge.html

et al., 2014a). Those knowledge bases have in common that they extract information from publicly available sources found on the web. Some of those might be more trustworthy than others, and there can be contradicting information on different sites. Additionally, the process of automatic extraction can lead to further errors (for example misrecognized characters when scanning documents) in the created knowledge base

- Information automatically gathered by sensors, like the status of services in information systems, weather data, or readings from industry process control systems. Sensors naturally have some margin of error that needs to be accounted for. Their readings (response time, temperature, pressure, ...) are different from information in text, and usually you have multiple values from the same source, contrary to one reading from multiple sources as with text.
- Data uncertain in its nature, like the occurrence of risks and threats in information systems, or forecasts of any kind. This data is inherently uncertain as it makes predictions about the future.

A number of approaches have been proposed for combining description logics with probabilistic or uncertain reasoning. An overview of early approaches is given by Lukasiewicz and Straccia (2008), more recent approaches include probabilistic description logics – e.g. DISPONTE/BUNDLE by Riguzzi et al. (2013, 2015) or Pronto by Klinov and Parsia (2013) – and Log-linear Description Logics by Niepert et al. (2011b). On the other hand, the logic programming and statistical relational learning community has developed probabilistic versions of Datalog-style languages – like ProbLog by De Raedt et al. (2007) – that can be used to partially model ontological background knowledge.

While for many of these languages efficient subsets have been identified, for example by Riguzzi et al. (2015) or Klinov and Parsia (2013), and optimized reasoning algorithms have been proposed, none of the existing approaches is designed to handle large amounts of data as we find on the Web, or it is unclear how well these approaches works in practice due to missing practical evaluation.

## 1.2. Research Objectives and Outline

The overall objective of this dissertation is to identify scenarios and datasets where methods that incorporate their inherent uncertainty improve results, and investigate approaches and tools that are suitable for the respective task. In summary, this work is set out to tackle the following objectives:

1. debugging uncertain knowledge bases in order to generate consistent knowledge graphs to make them accessible for logical reasoning,
2. combining probabilistic query answering and logical reasoning which in turn uses these consistent knowledge graphs to answer user queries, and

3. employing the aforementioned techniques to the problem of risk management in IT infrastructures, as a concrete real-world application.

In the following we outline the goals of the different chapters. Afterwards, we introduce description logic and probabilistic reasoning, and how both can be combined in one formalism.

## Debugging Large-scale Uncertain Temporal Knowledge Graphs

Uncertain knowledge graphs which were created automatically have an exceptionally high chance of containing contradicting or inconsistent information. This risk further increases when integrating data from multiple sources.

In Chapter 3 we look at the specific problem of debugging large-scale temporal information in uncertain knowledge graphs (containing hundreds of thousands or millions of statements). There is already a body of work in debugging and repairing regular knowledge graphs (cf. Fleischhacker (2014) for an overview). However, temporal data is special in the sense that it also needs reasoning for instants and intervals of time, for example the following two statements

$$\begin{gathered} \mathit{playsFor}(\mathit{CristianoRonaldo}, \mathit{Manchester}) \\ \mathit{playsFor}(\mathit{CristianoRonaldo}, \mathit{Chelsea}) \end{gathered}$$

contradict each other as a player can only play for a single team at a time, however, with the temporal context like

$$\begin{gathered} \mathit{playsFor}(\mathit{CristianoRonaldo}, \mathit{Manchester}, 2003, 2009) \\ \mathit{playsFor}(\mathit{CristianoRonaldo}, \mathit{Chelsea}, 2009, \textsc{Now}) \end{gathered}$$

they are valid.

## Scalable Probabilistic Query Answering and Logical Reasoning

The SPARQL recommendation[9] (The W3C SPARQL Working Group, 2013) provides an expressive query language for retrieving information from knowledge bases. Furthermore, ontology-based data access (ODBA) has received a lot of attention in the Semantic Web community. In particular, results on light-weight description logics that allow efficient reasoning and query answering provide new possibilities for using ontologies in data access. One approach for ontology-based data access is to rewrite a given query using the background ontology in such a way that the resulting – more complex – query can directly be executed on a relational database. This is possible for different light-weight ontology languages, in particular the *DL-Lite* family (Artale et al., 2009).

[9]`https://www.w3.org/standards/techs/sparql#stds`

Research in probabilistic databases has shown that there is a strict dichotomy of safe (data complexity in PTime) and unsafe (in #P-hard) queries (Suciu et al., 2011). For the probabilistic extension of OBDA the distinction between safe and unsafe queries is highly crucial. Jung et al. have shown that query rewriting for OBDA can directly be lifted to the probabilistic case (Jung and Lutz, 2012). Furthermore, they prove that the complexity results and the dichotomy of safe and unsafe queries also carries over to probabilistic query answering in an OBDA setting.

Chapter 4 analyzes the information needs of users sending queries to various public SPARQL endpoints, and how uncertain knowledge used to answer those queries can be processed efficiently. Furthermore, we investigate the performance of probabilistic OBDA both on real-world data and a benchmark dataset we created specifically for this task.

### IT Risk Management in Large-scale IT Infrastructures

In this chapter, we apply some of the aforementioned formalisms –especially ontologies and Markov logic networks – to the area of IT risk management. IT risk management tries to find, analyze and reduce risks in an IT infrastructure. Most commonly risk is defined as a set of triplets, each triplet consisting of a scenario, its potential impact, and its probability (Kaplan and Garrick, 1981). In the IT environment these scenarios are often also called threats. A very simple example would be the risk $\langle \mathit{HardriveCrash\text{-}Mailserver}, \mathit{Mail\text{-}Outage}, 1\% \rangle$.

If a new possible threat surfaces, the IT risk management needs to assess its probability and evaluate its potential impact. Today's IT has complex dependencies and a threat to a single component can threaten a whole infrastructure. Furthermore, single threats often have a very low probability but the combination of many threats can be a major risk to an IT infrastructure. Therefore, it is not enough to look at infrastructure components individually to determine the possible impact of a threat. Thus, a manual threat analysis takes a lot of time. However, a fast response to a new threat is important to minimize the chance of exploitation (Ernst & Young, 2012).

Additionally, once an outage occurs it is crucial to quickly identify the source of the problem. An efficient procedure for root cause analysis is thus an important tool in IT risk management. However, it suffers from the same problems as identifying and estimating threats in the first place.

In Chapter 5 we investigate the two scenarios of a) estimating the impact of risks on the availability of components in an infrastructure, and b) the task of root cause analysis. We propose and evaluate two novel approaches using Markov Logic Networks to solve these problems.

### Conclusion and Future Work

We finish with an overall conclusion in Chapter 6 of the dissertation. Afterwards we provide an outlook for future work in handling uncertain information in Chapter 7.

We have published parts of the research presented in this dissertation in the following workshop & conference papers and journal articles:

- Chapter 3 uses results from (Schoenfisch, 2014), (Schoenfisch and Stuckenschmidt, 2015), (Schoenfisch and Stuckenschmidt, 2016), and (Schoenfisch and Stuckenschmidt, 2017)
- Chapter 4 is based on (Chekol et al., 2017a) and (Chekol et al., 2017b)
- Chapter 5 on (von Stülpnagel et al., 2014), (Schoenfisch et al., 2016), and (Schoenfisch et al., 2017)

# 2 Preliminaries

In this chapter we introduce the underlying foundations for the remainder of this thesis. The ultimate idea of the approaches presented here is to provide a sound and complete translation of entailment in a logical formalism to a rule-based probabilistic formalism. This framework offers a high flexibility, as it can be used to combine a range of different logics and uncertainty formalisms.

Combining logics and uncertainty is an extensive and long standing field of research. In one of the earliest works, Nilsson (1986) proposes to assign probabilities instead of Boolean truth values to logical formulas. This generalizes ordinary logical entailment and provides a clear semantics for the entailment problem. However, a major problem is the complexity of reasoning which already makes entailment intractable for very small knowledge bases. To avoid this problem, other logical formalism have been considered, and a great number of different approaches to address uncertainty (e.g. fuzzy logic (Gerla, 1994), subjective logic (Jøsang, 2001), probabilistic graphical models (Koller and Friedman, 2009), probabilistic logic programming (Lukasiewicz, 1998; Ng and Subrahmanian, 1992), or evidential reasoning (Ruspini et al., 1992)) have been developed.

Recent approaches in the direction of probabilistic logic programming include Independent Choice Logic (Poole, 2008), PRISM (Sato and Kameya, 2008) and ProbLog (De Raedt et al., 2007). These approaches have been shown to be useful in practice, however, they suffer from the same problem of being hard to maintain as similar rule-based knowledge representations.

Probabilistic graphical models are another popular line of research, which has been proven to be useful in practical applications (Koller and Friedman, 2009). Template languages provide means to represent them in a compact way for complex domains, which lessens effort to create and maintain the. Usually, the template languages employ some first-order logic to represent probabilistic knowledge. By instantiating the first-order predicates with constants, the logical model is translated to a probabilistic graphical model. However, those approaches are not directly compatible with description logic and DL reasoning, as the templates are restricted to a finite domain of constants. One of the earliest approaches in this direction is *P-CLASSIC* (Koller et al., 1997) which extends the *Classic* description logic with probabilistic information. More recently, approaches like PR-OWL (da Costa and Laskey, 2006) and Markov logic networks (MLNs) (Richardson and Domingos, 2006) were developed. We will especially discuss MLNs below.

Lukasiewicz recently proposed approaches that combine probabilistic reasoning with description logic (Lukasiewicz, 2007) and probabilistic logic programs with ontological knowledge (Lukasiewicz, 2008). However, these approaches also suffer from a high complexity and a lack of tool support.

There is some work in the direction of combining light-weight description logic with Bayesian networks. $\mathcal{BEL}$ is a combination of the description logic $\mathcal{EL}$ and Bayesian networks (Ceylan and Peñaloza, 2017), however the formalism does not include uncertainty for instance data, and it is intractable. D'Amato et al. (2008) combine Bayesian networks with *DL-Lite*, and show that this approach offers satisfiability checking and query answering in LogSpace. However, when probabilistic information changes, the Bayesian network needs to be recalculated, which is a #P-hard problem.

In our work we investigate two different ways of combining light-weight description logics with probabilistic reasoning: Niepert et al. (2011b) proposed a log-linear description logic which combines $\mathcal{EL}$ with Markov logic networks. In their formalism, they employ research on consequence-driven reasoning for description logics (Kazakov, 2009; Krötzsch, 2010; Simancik et al., 2011). This approach is able to compute the most probable, coherent ontology from a given probabilistic knowledge base. The other approach, proposed by Jung and Lutz (2012), uses query rewriting in OWL 2 QL combined with traditional probabilistic reasoning. This promises scalable query answering for large probabilistic knowledge bases.

In the following, we first explain uncertain knowledge graphs and ontologies; common approaches to model knowledge bases. As formalisms for modeling them, we introduce first-order logic, especially description logic as a decidable fragment thereof, and in particular the two light-weight description logics $\mathcal{EL}^{++}$ and $\textit{DL-Lite}_R$. Then, we describe probabilistic reasoning – in particular Markov logic networks and probabilistic databases – and how to combine them with light-weight description logics to conduct probabilistic logical reasoning.

The formalisms and approaches described in this chapter are the foundation for the investigations which we will conduct in the following parts of this work. We first define the tools required to model (uncertain) information: knowledge graphs and ontologies. Then, we introduce first-order logic and light-weight fragments thereof, which are used for inference and reasoning. Subsequently, we describe how light-weight description logics are combined with probabilistic reasoning formalisms to conduct probabilistic logical reasoning.

## 2.1. Knowledge Graphs

A knowledge graph is a directed graph whose nodes represent instances of objects in the real world referenced by IRIs (Internationalized Resource Identifiers (Duerst and

Suignard, 2005)), which are connected via properties to other instances or literals. Popular examples of knowledge graphs are YAGO (Hoffart et al., 2013), Google Knowledge Vault (Dong et al., 2014b), DBpedia (Lehmann et al., 2015) or Wikidata (Lehmann et al., 2015).

Usually, the graph is defined by statements in the form of subject-predicate-object triples. More formally:

**Definition 2.1** (Knowledge Graph)
*Let $\mathcal{I}$, $\mathcal{P}$, and $\mathcal{L}$ be disjoint sets with $\mathcal{I}$ containing IRIs identifying real-world objects, $\mathcal{P}$ consisting of IRIs representing relations and properties, and $\mathcal{L}$ being literals. A knowledge graph $\mathcal{K}$ is then the set $\{(s,p,o) : (s,p,o) \in \mathcal{I} \times \mathcal{P} \times \mathcal{I} \cup \mathcal{L}\}$[1].* ○

For example a statement about Claudio Ranieri being coach of Chelsea F.C. looks as following in the Wikidata knowledge graph[2]:

**Example 2.1** (Knowledge Graph)

$$(\mathit{wd}{:}\ \mathit{Q235068}, \mathit{wdt}{:}\ \mathit{P286}, \mathit{wd}{:}\ \mathit{Q9616})$$
$$(\mathit{Q235068}, \mathit{wikibase}{:}\ \mathit{label}, \mathit{Claudio\ Ranieri})$$
$$(\mathit{P286}, \mathit{wikibase}{:}\ \mathit{label}, \mathit{head\ coach})$$
$$(\mathit{Q9616}, \mathit{wikibase}{:}\ \mathit{label}, \mathit{Chelsea\ F.\ C.})$$

*Note how Wikidata also employs the common technique of assigning globally unique IDs to instances and properties and defining human-readable labels as additional statements. This avoids possible conflicts between entities having the same name, for example consider* Berlin*, which is the capital of Germany, but also the name of over 20 cities in the United States of America.* △

### 2.1.1. Uncertain Knowledge Graphs

A knowledge graph can be turned into an uncertain knowledge graph by annotating the triples with an element that expresses uncertainty. Those elements can be based on any formalism for representing uncertainty, for example the well-known probability theory (Mikosch and Kallenberg, 1998), fuzzy sets (Zadeh, 1965), possibility theory (Dubois and Prade, 2001), or weighted models like log-linear models as used in Markov logic networks (cf. Section 2.4). Similar to knowledge graphs, we define uncertain knowledge graphs as follows:

[1]RDF has the notion of *blank nodes*, i.e., nodes without a named, distinct identity. Those are not considered in this work.

[2]For brevity and readability we use the namespace prefixes as defined by Wikidata here: `https://www.mediawiki.org/wiki/Wikibase/Indexing/RDF_Dump_Format#Prefixes_used`

**Definition 2.2** (Uncertain Knowledge Graphs)
*Let $\mathcal{K}$ be knowledge graph as defined above, and $\mathcal{W}$ be a set of elements expressing degrees of uncertainty. A uncertain knowledge graph $\mathcal{K}_{\mathcal{W}}$ is then the set $\{(s,p,o,w) : (s,p,o) \in \mathcal{K}, w \in \mathcal{W}\}$.* ○

## 2.2. Ontologies and First-order Logic

An ontology can be understood as a logical representation of a domain model. A knowledge graph combined with logical rules is one concrete of many possible forms of an ontology. The advantages of such domain models include enabling the sharing of knowledge, the re-use of knowledge, and the better engineering of knowledge-based systems with respect to acquisition, verification and maintenance (Jones et al., 1998).

There are several logical languages that can be used to define ontologies. One of the most important formalisms is the family of description logics (DL) (Baader and Nutt, 2003), a fragment of first-order logic, which is based on a well-defined model-theoretic semantics. The core reasoning problems for DL languages are (usually) decidable, and efficient decision procedures have been designed.

A logical formalization that is built on top of a well-defined semantics has several advantages. One of these advantages is the possibility to exploit reasoning capabilities. Reasoning can be used to detect inconsistencies in the model. This helps both the task of acquisition and maintenance by automatically detecting potential mistakes. Furthermore, reasoning is used to make information explicit that is only stated implicitly in the model. For example, when we know that every parent is a person, and Alice is a parent, the implicit information is that Alice is also a person.

### 2.2.1. First-order Logic and Description Logic

#### First-order Logic

First-order logic (FOL) is used do describe and reason in a domain of discourse. Syntactically, it consists of:

*constants* $\mathcal{C} = \{c_1, ..., c_{|\mathcal{C}|}\}$,

*variables* $\mathcal{V} = \{v_1, ..., v_{|\mathcal{V}|}\}$,

*predicates* $\mathcal{R} = \{r_1, ..., r_{|\mathcal{R}|}\}$,

*functions* $\mathcal{F} = \{f_1, ..., f_{|\mathcal{F}|}\}$,

and *logical operators* $(\forall, \exists, \wedge, \vee, \rightarrow, \leftrightarrow, \neg, (,), \equiv)$.

A *term t* can either be a variable $v$, a constant $c$, or a function of terms $f(t_i, ..., t_j)$. An *atomic formula* (or simply atom) is a formula that contains no logical connective, i.e. it consists of a single predicate. A general formula is created by connecting multiple atoms via logical operators (using the usual semantics in predicate logic as given by Hilber and Ackermann (1938)). If an atom contains no variables we call it a *ground atom*; if a formula contains no free variables (i.e. only constants and variables bound by $\forall$ or $\exists$), we call it a *ground formula*.

The semantics of a first-order logic is given by an *interpretation*. Intuitively, an interpretation assigns real-world objects present in the domain to the syntactic constructs described above. This way, every term is also assigned a truth value, e.g. an atomic formula is true if a relation as identified by the predicate exists (or can exist) between the specified constants and variables.

A small example FOL model is the following simple description of relations between persons and their hobbies:

**Example 2.2** (First-order Logic Model)

$$\mathit{person}(Alice) \qquad \mathit{person}(Bob) \qquad \mathit{person}(Eve)$$

$$\begin{gathered}\mathit{friends}(Alice, Bob)\\ \mathit{hasHobby}(Alice, Football)\end{gathered}$$

$$\mathit{friends}(x, y) \land \mathit{hasHobby}(x, z) \rightarrow \mathit{hasHobby}(y, z)$$

*The model contains the constants* Alice*,* Bob *and* Eve*, variables $x, y$ and $z$, the predicates* friends *and* hasHobby *and the logical operators $\land$ and $\rightarrow$. It states that* Alice *and* Bob *are both persons and friends,* Eve *is a person,* Alice *has* Football *as a hobby, and that individuals who are friends have the same hobbies. From this model it can be inferred that* Bob *also has the hobby* Football*.* $\triangle$

## Description Logic

Description logic (DL) is a fragment of first-order logic. Its expressive power is usually between simple propositional logic and full first-order logic. Description logics are mainly created for knowledge representation, and the possible syntactically allowed constructs are carefully chosen to enable efficient reasoning.

Description logics use a different terminology from FOL: constants are either *concepts* or *instances* thereof, and predicates are denoted as *roles*. If a concept or a role is constructed from other (atomic) concepts or roles, it is called a *complex concept* or *role*, respectively. DLs also introduce two special concepts: $\top$ (top) with every individual as its instance, and $\bot$ with no individual as instance. A *concepts assertion* or a *role* *assertion* is the instantiation of a concept with an individual or individuals, respectively. Usually, the set of concepts and roles is denoted as TBox (terminological box, or simply terminology), and the set of individuals and assertions as ABox (assertional box, or also world description). We define TBox and ABox in more detail below.

A simple description logic, which is the basis for many more expressive DLs is $\mathcal{ALC}$ (*attributive language with complex concept negation*) (Baader and Nutt, 2003):

**Definition 2.3** (The Description Logic $\mathcal{ALC}$)
*Let $A$ be an arbitrary atomic concept, $C$ and $D$ arbitrary complex concepts, $a$ and $b$ arbitrary individuals, and $R$ be some role. Then, the syntax to build constructs in $\mathcal{ALC}$ is as follows:*

$$\begin{aligned} C, D \longrightarrow \quad & A \;\mid\; \neg A \;\mid\; \neg C \;\mid \\ & \top \;\mid\; \bot \;\mid \\ & C \sqcap D \;\mid \\ & \forall R.C \;\mid \\ & \exists R.\top \end{aligned}$$

*The formal semantics of those concepts are given by an interpretation $\mathcal{I}$ over a non-empty domain $\Delta^{\mathcal{I}}$ and an interpretation function $\cdot^{\mathcal{I}}$, which assigns every atomic concept $A$ to a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, and every atomic role $R$ to a set $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. For complex concepts, the interpretation function is as follows:*

$$\begin{aligned} \top^{\mathcal{I}} &= \Delta^{\mathcal{I}} \\ \bot^{\mathcal{I}} &= \emptyset \\ A^{\mathcal{I}} &\subseteq \Delta^{\mathcal{I}} \\ R^{\mathcal{I}} &\subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \\ (\neg A)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \backslash A^{\mathcal{I}} \\ (C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}} \\ (\forall R.C)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid \forall b.(a,b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\} \\ (\exists R.\top)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid \exists b.(a,b) \in R^{\mathcal{I}} \rightarrow b \in \Delta^{\mathcal{I}}\} \end{aligned}$$

○

**Terminologies** *Terminologies* (or *TBox* in short) define relationships between concepts and roles. Terminological axioms are of the form

$$\begin{aligned} C &\sqsubseteq D & \qquad C &\equiv D \\ R &\sqsubseteq S & \qquad R &\equiv S \end{aligned}$$

where $C$ and $D$ are concepts, and $R$ and $S$ are roles. Axioms using the $\sqsubseteq$-operator are called *inclusions*, whereas those using $\equiv$ are called *equalities*. Again, the semantics of those axioms are defined by an interpretation $\mathcal{I}$. $\mathcal{I}$ *satisfies* an inclusion $C \sqsubseteq D$ if $C^{\mathcal{I}} \subset D^{\mathcal{I}}$, and an equality $C \equiv D$ if $C^{\mathcal{I}} = D^{\mathcal{I}}$. The same is analogous for roles. An interpretation $\mathcal{I}$ is called a *model* for a set of axioms if it satisfies all of them. If two sets of axioms have the same models, they are *equivalent*.

**World Description** After defining a terminology, we want to describe our world or domain of interest using this vocabulary. This world descriptions is called the *ABox*. In the ABox, individuals are given properties by asserting them to concepts or roles. For individuals $a$, $b$, and $c$, those are of the form

$$C(a) \qquad\qquad R(b, c)$$

$C(a)$ is called a *concept assertion*, denoting that $a$ is of type $C$, and $R(a, b)$ is called *role assertion*, relating $a$ to $b$ via the given role.

The semantics of the ABox are given by extending the interpretation $\mathcal{I}$ to also map to each individual $a$ an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. Throughout this work, we make the *unique name assumption* (UNA), i.e. for two individuals $a$ and $b$ with distinct names, $a^{\mathcal{I}} \neq b^{\mathcal{I}}$. $\mathcal{I}$ satisfies a concept assertion $C(a)$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$, and a role assertion $R(a, b)$ if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$. Analogous to the TBox, $\mathcal{I}$ satisfies the ABox and is a model for it, if it satisfies every assertion of the ABox.

With this simple description logic, we can formulate a knowledge base similar to the one in Example 2.2:

**Example 2.3** (Description Logic Model)

$$\begin{gathered} \mathit{Man} \sqsubseteq \mathit{Person} \\ \mathit{Woman} \sqsubseteq \mathit{Person} \\ \mathit{Woman} \sqcap \mathit{Man} \sqsubseteq \bot \\ \mathit{Friend} \equiv \exists hasFriend.\top \end{gathered}$$

$$\begin{gathered} \mathit{Woman}(\mathit{alice}) \quad \mathit{Man}(\mathit{bob}) \quad \mathit{Woman}(\mathit{eve}) \\ \mathit{Hobby}(\mathit{football}) \\ \mathit{Friends}(\mathit{alice}, \mathit{bob}) \quad \mathit{HasHobby}(\mathit{alice}, \mathit{football}) \end{gathered}$$

*We extended the model with the concepts of* Man*,* Woman *and* Friend*, and formulas stating that all men and women are persons, and that there are no instances which are both man and woman. We also state that a* Friend *is somebody who has a friend. However, we cannot model the rule* $\mathit{friends}(x, y) \wedge \mathit{hasHobby}(x, z) \rightarrow \mathit{hasHobby}(y, z)$*, as rules are not part of standard description logics.*

*Note that concepts and roles in description logics are usually starting with an uppercase and instances with a lowercase letter.* △

**Nominals** It is also possible to use individual names directly in the TBox. They are used to enumerate individuals which form a concept. For example the concept of the three primary colors can be defined as follows:

$$\text{PrimaryColors} \equiv \{\text{red}, \text{yellow}, \text{blue}\}$$

Individual names inside the TBox are called *nominals*.

**Ontologies** The TBox containing concepts, roles, inclusions, and equalities, together with the ABox consisting of concept and role assertions, form an *ontology*, often denoted as $\mathcal{K} = \{\mathcal{T}, \mathcal{A}\}$[3].

### Inference in Description Logics

The main advantage of ontologies is the ability to ensure consistency of the modeled knowledge, and to deduce new knowledge form existing information. Through *logical inference* we can, for example, reason from Example 2.3 that *Alice*, *Bob*, and *Eve* are all persons.

In description logics there exist four different kinds of reasoning tasks regarding the TBox (Baader et al., 2003):

- **Satisfiability:** A concept $C$ is *satisfiable* with respect to $\mathcal{T}$ if there exists a model $\mathcal{I}$ of $\mathcal{T}$ such that $C^{\mathcal{I}}$ is nonempty. In this case we say also that $\mathcal{I}$ is a model of $C$.
- **Subsumption:** A concept $C$ is subsumed by a concept $D$ with respect to $\mathcal{T}$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{T}$. In this case we write $C \sqsubseteq_{\mathcal{T}} D$ or $\mathcal{T} \models C \sqsubseteq D$.
- **Equivalence:** Two concepts $C$ and $D$ are equivalent with respect to $\mathcal{T}$ if $C^{\mathcal{I}} = D^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{T}$. In this case we write $C \equiv_{\mathcal{T}} D$ or $\mathcal{T} \models C \equiv D$.
- **Disjointness:** Two concepts C and D are disjoint with respect to $\mathcal{T}$ if $C^{\mathcal{I}} \cap D^{\mathcal{I}} = \emptyset$ for every model $\mathcal{I}$ of $\mathcal{T}$.

Those reasoning tasks also straightforwardly transfer to roles. If all concepts $C \in \mathcal{T}$ are satisfiable, i.e. $C \not\sqsubseteq_{\mathcal{T}} \bot$ or $\mathcal{T} \models C \not\sqsubseteq \bot$, we also say $\mathcal{T}$ is *coherent*, otherwise we call it *incoherent*.

**Example 2.4** (Incoherent TBox)
*Consider the TBox in Example 2.3 extended with the following axioms:*

$$\begin{aligned} &\mathit{Homunculus} \sqsubseteq \mathit{Woman} \\ &\mathit{Homunculus} \sqsubseteq \mathit{Man} \end{aligned}$$

[3]Note that ontologies need not be modeled with description logics, although the use of description logics is very common

*Those axioms define that every individual in the concept* Homunculus *would be both a* Man *and a* Woman. *However, these concepts are defined to be disjoint. Thus, we can infer the following:*

$$\textit{Homunculus} \sqsubseteq \textit{Woman} \sqcap \textit{Man} \sqsubseteq \bot$$

*Consequently, as* Homunculus *is unsatisfiable, the TBox is incoherent.* $\triangle$

Considering the ABox, there are two additional reasoning tasks (Baader et al., 2003):

- **Consistency:** An ABox $\mathcal{A}$ is consistent with respect to a TBox $\mathcal{T}$, if there is an interpretation that is a model of both $\mathcal{A}$ and $\mathcal{T}$.
- **Instance Checking:** We say that an assertion $\alpha$ is entailed by $\mathcal{A}$ and write $\mathcal{A} \vDash \alpha$, if every model of $\mathcal{A}$, also satisfies $\alpha$.

We say that an ABox $\mathcal{A}$ is *consistent* with respect to $\mathcal{T}$, if there is a model for both $\mathcal{T}$ and $\mathcal{A}$.

**Example 2.5** (Inconsistent ABox and TBox)
*Again consider the axioms given in Example 2.3. Now assume we add the following ABox axioms:*

$$\textit{Woman}(\textit{robin}) \qquad \textit{Man}(\textit{robin})$$

*From these assertions, we can infer that robin* $\in \textit{Woman}^{\mathcal{I}} \cap \textit{Man}^{\mathcal{I}} = \emptyset$*. This is obviously a contradiction, making the ABox inconsistent regarding the TBox.* $\triangle$

**Open World vs Closed World Assumption** The TBox and ABox of an ontology can roughly be related to the schema and the data in a relational database. Apart from (description) logic being more expressive than relational algebra, there is another important difference: the semantics for absent information. An ontology has many different models, whereas a relation database has exactly the one given by the data. Consider for example Example 2.3: In this knowledge base, we explicitly know of the three persons *Alice*, *Bob*, and *Eve*. If this information interpreted as ontology and we want to check whether the individual *Dave* is a person, the answer is *unknown*; there is no information supporting or contradicting it. Now, suppose we have this knowledge in a relational database and again want to know if *Dave* is a person. The answer would clearly be *no*.

These two different semantics for absent information are called *open world assumption* and *closed world assumption*. Under the open world assumption, one cannot infer true or false statements from missing data. Opposite to that, under the closed world assumption, the known world is considered to be complete, thus missing data is interpreted as false.

Ontology languages and reasoning procedures usually adopt the open world assumption. However, reasoning with an open world assumption becomes more complex. Because of this, and as we are mainly concerned with query answering where *unknown* is a very unsatisfying answer for a user, we generally restrict reasoning to the closed world assumption throughout this work.

## 2.3. Light-weight Description Logics

In this section we describe light-weight description logics, which we will use throughout this work. Light-weight description logics are carefully crafted families of logics to allow for tractable reasoning, i.e. the reasoning tasks can be decided in at most polynomial time (in contrast to, for example $\mathcal{ALC}$ as introduced earlier, which requires exponential time in the worst case). The two light-weight description logics on which we focus in this work are $\mathcal{EL}^{++}$ (Baader et al., 2005) and *DL-Lite*$_R$ Calvanese et al. (2005).

### 2.3.1. The Description Logic $\mathcal{EL}^{++}$

The description logic $\mathcal{EL}^{++}$ is designed for efficient reasoning with large TBoxes. It is the basis for the OWL 2 EL profile[4].

**Definition 2.4** (The Description Logic $\mathcal{EL}^{++}$)
*The syntax of $\mathcal{EL}^{++}$ is as follows:*

$$\begin{aligned} C, D &\longrightarrow A \mid \{a\} \mid \top \mid \bot \mid \\ &\phantom{\longrightarrow} C \sqcap D \mid \exists R.C \\ R, S &\longrightarrow R \mid R_1 \circ \ldots \circ R_n \end{aligned}$$

$$\begin{aligned} &C \sqsubseteq D && C \equiv D \\ &R \sqsubseteq S && R \equiv S \end{aligned}$$

*where $A$ is an atomic concept, $C$ and $D$ are complex concepts, $a$ is an individual, and $R$ and $S$ are roles.*

*The semantics are as given in Definition 2.3, with the following addition:*

$$\begin{aligned} \{a\}^{\mathcal{I}} &= \{a^{\mathcal{I}}\} \\ (\exists R.C)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid \exists b.(a,b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\} \\ (R_1 \circ \ldots \circ R_n \sqsubseteq R)^{\mathcal{I}} &= R_1^{\mathcal{I}} \circ \ldots \circ R_n^{\mathcal{I}} \sqsubseteq R^{\mathcal{I}} \end{aligned}$$

*Given a TBox $\mathcal{T}$, we denote $\mathsf{BC}_{\mathcal{T}}$ as the set of* basic concept descriptions*, i.e., the smallest set of concept descriptions consisting of $\top$ and all concept names and nominals appearing in $\mathcal{T}$. An $\mathcal{EL}^{++}$ ontology is in* normalized *if all axioms have the following form, where $A, B \in \mathsf{BC}_{\mathcal{T}}$ and $C \in \mathsf{BC}_{\mathcal{T}} \cup \{\bot\}$:*

$$\begin{aligned} A &\sqsubseteq C & A \sqcap B &\sqsubseteq C \\ \exists r.A &\sqsubseteq C & A &\sqsubseteq \exists r.B \\ R_1 &\sqsubseteq R_2 & R_1 \circ R_2 &\sqsubseteq R \end{aligned}$$

[4]`https://www.w3.org/TR/owl2-profiles/#OWL_2_EL`

*By applying a finite set of rules and introducing new concept and role names, any knowledge base $\mathcal{K} = \{\mathcal{T}, \mathcal{A}\}$ can be turned into a normalized knowledge base of size polynomial in $\mathcal{K}$ (Baader et al., 2005). For the remainder of this work we always assume $\mathcal{K}$ to be normalized.* ○

### 2.3.2. The Description Logic *DL-Lite*$_R$

The description logic *DL-Lite*$_R$ is designed for large ABoxes and efficient query answering. The OWL 2 QL profile uses this logic as its foundation[5].

**Definition 2.5** (The Description Logic *DL-Lite*$_R$ (Calvanese et al., 2007))

$$\begin{aligned} B &\rightarrow A \mid \exists R \mid A \sqcap A' & \qquad C &\rightarrow B \mid \neg B \\ R &\rightarrow P \mid P^- \mid P \sqcap P' & \qquad S &\rightarrow R \mid \neg R \end{aligned}$$

$$\begin{array}{cc} B \sqsubseteq C & \qquad R \sqsubseteq S \\ A(a) & \qquad P(a,b) \end{array}$$

*where $A$ denotes an atomic concept, $P$ an atomic role, and $P^-$ the inverse of the atomic role $P$. $B$ denotes a basic concept, i.e. either an atomic concept or a concept of the form $\exists R$, where $R$ denotes a basic role, that is, either an atomic role or the inverse of an atomic role. $C$ denotes a complex concept, which can be a basic concept or its negation, and $S$ denotes a complex role, which can be a basic role or its negation.*

*The semantics of a DL is as an interpretation $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$, consisting of a non-empty interpretation domain $\Delta^\mathcal{I}$ and an interpretation function $\cdot^\mathcal{I}$ that assigns to each concept $C$ a subset $C^\mathcal{I}$ of $\Delta^\mathcal{I}$, and to each role $R$ a binary relation $R^\mathcal{I}$ over $\Delta^\mathcal{I}$:*

$$\begin{aligned} A^\mathcal{I} &\subseteq \Delta^\mathcal{I} \\ P^\mathcal{I} &\subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I} \\ (P^-)^\mathcal{I} &= \{(o_2, o_1) \mid (o_1, o_2) \in P^\mathcal{I}\} \\ (\exists R)^\mathcal{I} &= \{o \mid \exists o'.(o, o') \in R^\mathcal{I}\} \\ (\neg A)^\mathcal{I} &= \Delta^\mathcal{I} \setminus A^\mathcal{I} \\ (\neg R)^\mathcal{I} &= \Delta^\mathcal{I} \times \Delta^\mathcal{I} \setminus R^\mathcal{I} \end{aligned}$$

$$\begin{aligned} a^\mathcal{I} &= a \in \Delta^\mathcal{I} \\ (C(a))^\mathcal{I} &= a \in C^\mathcal{I} \\ (P(a,b))^\mathcal{I} &= (a,b) \in P^\mathcal{I} \end{aligned}$$

○

[5] `https://www.w3.org/TR/owl2-profiles/#OWL_2_QL`

## 2.4. Probabilistic Graphical Models

Probabilistic graphical models represent complex distributions over multiple random variables as a graph-based structure (Koller and Friedman, 2009). The most well-known probabilistic graphical model are Bayesian networks, which use a directed acyclic graph as the foundation to described the dependencies of random variables and their distribution. Opposed to that, the models we consider for this work are based on undirected graphs, which are better suited for cycles and symmetric relations, common in knowledge bases.

### Markov Logic Networks

Markov Logic Networks (MLN) generalize probabilistic graphical and first-order logic models by allowing hard and soft first-order formulas (Richardson and Domingos, 2006). Hard formulas are regular first-order formulas, which have to be fulfilled by every interpretation. An interpretation is also referred to as a possible world. Soft formulas have weights that support (in case of positive weights) or penalize (in case of negative weights) worlds in which they are satisfied. The probability of a possible world, one that satisfies all hard formulas, is proportional to the exponential sum of the weights of the soft formulas that are satisfied in that world. This corresponds to the common understanding of Markov Networks as log-linear probabilistic models (Richardson and Domingos, 2006).

MLNs are a template for constructing Markov Networks. A formula is called a grounded formula if all variables have been replaced by constants. Given a set of constants, a Markov Network can be generated from the MLN by computing all possible groundings of the given formulas. Due to the closed world assumption, the domain of interest consists of only those entities that are defined by specifying the set of constants. An atom is a formula that consists of a single predicate. A possible world corresponds to a set of ground atoms, which is usually a small subset of all possible groundings.

Furthermore, in some implementations for inference predicates can be defined as being either observed or hidden (unobserved) and get assigned some type. For an observed predicate, only explicitly stated groundings are allowed and considered to be true, whereas for hidden predicates, any grounding with the given constants can be generated. Typing allows to restrict the possible constants for grounding a predicate to subset. Both techniques are used to restrict the number of possible groundings and thus the effort needed for inference.

**Example 2.6**
*Revisiting the previous example about persons and hobbies, if all predicates were untyped*

*and hidden the following is an excerpt of a possible world:*

$$\begin{aligned} &\mathit{person}(\mathit{Alice}) && \mathit{person}(\mathit{Bob}) \\ &\mathit{person}(\mathit{Eve}) && \mathit{person}(\mathit{Football}) \\ &\mathit{friends}(\mathit{Alice}, \mathit{Alice}) && \mathit{friends}(\mathit{Alice}, \mathit{Bob}) \\ &\mathit{friends}(\mathit{Alice}, \mathit{Eve}) && \mathit{friends}(\mathit{Alice}, \mathit{Football}) \\ &\dots \\ &\mathit{hasHobby}(\mathit{Alice}, \mathit{Football}) && \mathit{hasHobby}(\mathit{Alice}, \mathit{Bob}) \\ &\dots \end{aligned} \tag{2.1}$$

*With no further restrictions,* Alice*,* Bob*,* Eve *and* Football *would all be persons, friends and have each other as hobby. Some of those groundings, like hasFriend(*Alice*,* Football*), do generally not make sense. By restricting* person *and* friends *to be observed and typed to only persons, and* hasHobby *to be typed to* person *and the newly introduced predicate* hobby*, the same possible world is reduced to this:*

$$\begin{aligned} &\mathit{person}(\mathit{Alice}) && \mathit{person}(\mathit{Bob}) \\ &\mathit{person}(\mathit{Eve}) && \mathit{hobby}(\mathit{Football}) \\ &\mathit{friends}(\mathit{Alice}, \mathit{Bob}) && \mathit{hasHobby}(\mathit{Alice}, \mathit{Football}) \\ &\mathit{hasHobby}(\mathit{Bob}, \mathit{Football}) && \mathit{hasHobby}(\mathit{Eve}, \mathit{Football}) \end{aligned} \tag{2.2}$$

Eve *still has the hobby* Football *despite having no friends, but the overall number of groundings is greatly reduced. By making the implication* $\text{friends}(x, y) \wedge \text{hasHobby}(x, z) \rightarrow \text{hasHobby}(y, z)$ *a soft rule with positive weight and adding the rules* $\text{friends}(x, y)$ *and* $\text{hasHobby}(x, y)$ *with a small negative weight we can also discourage worlds with additional* friends *and* hasHobby *groundings, except if they are enforced by the implication (cf. also Example 2.7 below for the complete formal definition of this example).* △

Formally, an MLN $L$ is a set of pairs $\langle F_i, w_i \rangle$, where $F_i$ is a first-order logic formula and $w_i$ is a real numbered weight. The MLN $L$, combined with a finite set of constants $\mathcal{C} = \{c_1, c_2, \dots, c_{|\mathcal{C}|}\}$, defines a ground Markov Network $M_{L,\mathcal{C}}$ as follows (Richardson and Domingos, 2006, p. 113):

1. $M_{L,\mathcal{C}}$ has one binary node for each possible grounding of each predicate in $L$. The value of the node is 1 if the grounded atom is true and 0 otherwise.
2. $M_{L,\mathcal{C}}$ contains one feature for each possible grounding of each formula $F_i$ in $L$. The value of this feature is 1 if the formula is true, and 0 otherwise. The weight of the feature is the $w_i$ associated with $F_i$ in $L$.

Generally, a feature can be any real-valued function of the variables of the network. In this work we use binary features, essentially making the value of the function equal to the truth value of the grounded atom.

The description as a log-linear model leads to the following definition for the probability distribution over possible worlds $x$ for the Markov Network $M_{L,C}$:

$$P(X=x) = \frac{1}{Z} \exp\left(\sum_i w_i n_i(x)\right) \tag{2.3}$$

where Z is a normalization constant and $n_i(x)$ is the number of true groundings of $F_i$ in $x$.

When describing the MLN we use the format ⟨*first-order formula*, *weight*⟩. Hard formulas have infinite weights. If the weight is $+\infty$ the formula must always be true, if the weight is $-\infty$ it must always be false. A soft formula with weight 0 has equal probabilities for being satisfied or not. We can also specify a fully grounded formula, which is considered as evidence or observations of the real world. To optimize the grounding step and reduce the amount of possible worlds, we also allow to specify the set of predicates together with the types that they use like this {predicate$_1$(*Type$_1$*), predicate$_1$(*Type$_2$*), predicate$_3$(*Type$_1$*, *Type$_2$*)}.

**Example 2.7** (Markov Logic Network)
*The formal definition of the predicates and the MLN for the example above then looks as following:*

$$\begin{gathered}\{\ \mathit{person}(\mathit{Person}), \mathit{friends}(\mathit{Person}, \mathit{Person}),\\ \mathit{hobby}(\mathit{Hobby}), \mathit{hasHobby}(\mathit{Person}, \mathit{Hobby})\ \}\\ \langle \mathit{friends}(x,y) \land \mathit{hasHobby}(x,z) \rightarrow \mathit{hasHobby}(y,z), 2\rangle\\ \langle \mathit{friends}(x,y), -0.1\rangle \qquad \langle \mathit{hasHobby}(x,y), -0.1\rangle\end{gathered} \tag{2.4}$$

*The evidence is formalized like this:*

$$\begin{gathered}\langle \mathit{person}(\mathit{Alice}), +\infty\rangle \qquad \langle \mathit{person}(\mathit{Bob}), +\infty\rangle\\ \langle \mathit{person}(\mathit{Eve}), +\infty\rangle \qquad \langle \mathit{hobby}(\mathit{Football}), +\infty\rangle\\ \langle \mathit{friends}(\mathit{Alice}, \mathit{Bob}), 3.5\rangle\\ \langle \mathit{hasHobby}(\mathit{Alice}, \mathit{Football}), 1.8\rangle\end{gathered} \tag{2.5}$$

△

## 2.5. Probabilistic Reasoning with Light-weight Description Logics

Finally, in this section we describe in detail two different approaches and formalism to combine probabilistic reasoning with description logics. In particular, we explain *log-linear description logic* and *tuple-independent OWL*. We then evaluate these two formalisms in different scenarios throughout this work.

### 2.5.1. Log-linear Description Logics

Log-linear description logics were introduced by (Niepert et al., 2011b) as a combination of the light-weight description logic $\mathcal{EL}^{++}$ (Section 2.3.1) and Markov logic networks (Section 2.4). They call this combination $\mathcal{EL}^{++}$-LL.

**Definition 2.6** (The Log-linear description logic $\mathcal{EL}^{++}$-LL (Niepert et al., 2011b))
*The syntax of $\mathcal{EL}^{++}$-LL is equivalent to that of $\mathcal{EL}^{++}$, with the addition of the possibility to assign weights to concept and role inclusions. More formally, a $\mathcal{EL}^{++}$-LL knowledge base $\mathcal{K}$ is a pair $(\mathcal{K}^D, \mathcal{K}^U)$ consisting of the deterministic $\mathcal{EL}^{++}$ KB $\mathcal{K}^D = d_i$ with some axioms $d_i$ and an uncertain KB $\mathcal{K}^U = (u_i, w_{u,i})$ which is a set of pairs of $\mathcal{EL}^{++}$ axioms $u_i$ and real-valued weights $w_{u,i}$, with $d_i \cap u_i \equiv \emptyset$.*

*The semantics of a $\mathcal{EL}^{++}$-LL knowledge base are defined by a joint probability distribution over the* coherent *KB. Given a $\mathcal{EL}^{++}$-LL knowledge base $\mathcal{K} = (\mathcal{K}^D, \mathcal{K}^U)$, the probability of a knowledge base $\mathcal{K}'$ over the same axioms is as follows:*

$$P(\mathcal{K}') = \begin{cases} \frac{1}{Z} \exp\left( \sum_{(u,w_u)\in\mathcal{K}^U : \mathcal{K}' \models u} w_u \right) & \text{if } \mathcal{K}' \text{ is coherent and } \mathcal{K}' \models \mathcal{K}^D \\ 0 & \text{otherwise} \end{cases} \tag{2.6}$$

*where $Z$ is the normalization constant of the log-linear probability distribution $P$.* ○

From this definition we have that an $\mathcal{EL}^{++}$-LL axiom is either part of the deterministic KB and definitely true, or part of the uncertain KB with a weight attached. Intuitively, the greater the weight of an axiom the more likely it is true (respectively false, if the weight is negative). Contrary to standard EL++, in $\mathcal{EL}^{++}$-LL we assume finite sets of concept, role, and individual names.

Next, we need a translation of the $\mathcal{EL}^{++}$-LL knowledge base to Markov logic networks, in order to compute the most probable, coherent ontology.

**Definition 2.7** (Translating $\mathcal{EL}^{++}$-LL into Markov Logic Networks)
*A probabilistic $\mathcal{EL}^{++}$-LL TBox is translated into the first-order logic formulas in a Markov logic network by applying the following mappings (Niepert et al., 2011b):*

$$\begin{array}{lcl} A \sqsubseteq C & \mapsto & sub(A, C) \\ A \sqcap B \sqsubseteq C & \mapsto & int(A, B, C) \\ A \sqsubseteq \exists R.B & \mapsto & rsup(A, R, B) \\ \exists R.A \sqsubseteq C & \mapsto & rsub(A, R, C) \\ R_1 \sqsubseteq R_2 & \mapsto & psub(R_1, R_2) \\ R_1 \circ R_2 \sqsubseteq R & \mapsto & pcom(R_1, R_2, R) \end{array}$$

*As the following equalities hold in $\mathcal{EL}^{++}$-LL*

$$C(a) \Leftrightarrow \{a\} \sqsubseteq C$$
$$R(a,b) \Leftrightarrow \{a\} \sqsubseteq \exists R.b$$

$$\langle C(a), w_1\rangle \Leftrightarrow \langle \{a\} \sqsubseteq C, w_1\rangle$$
$$\langle R(a,b), w_2 \Leftrightarrow \langle \{a\} \sqsubseteq \exists R.b, w_2\rangle$$

*we transform the ABox by formulating instances as nominals and employing the same mappings as above* ○

In order to conduct reasoning in $\mathcal{EL}^{++}$-LL we transform the inference rules for $\mathcal{EL}^{++}$ in a similar way:

**Definition 2.8** (Inference in $\mathcal{EL}^{++}$-LL)
*Inference in $\mathcal{EL}^{++}$-LL is implemented by formulating the completion rules given by Baader et al. (2005) into first-order logic formulas of a Markov logic network in the following way (Niepert et al., 2011b):*

($F_1$) $\forall c : sub(c,c)$
($F_2$) $\forall c : sub(c,\top)$
($F_3$) $\forall c,c',d : sub(c,c') \wedge sub(c',d) \Rightarrow sub(c,d)$
($F_4$) $\forall c,c_1,c_2,d : sub(c,c_1) \wedge sub(c,c_2) \wedge int(c_1,c_2,d) \Rightarrow sub(c,d)$
($F_5$) $\forall c,c',r,d : sub(c,c') \wedge rsup(c',r,d) \Rightarrow rsup(c,r,d)$
($F_6$) $\forall c,r,d,d',e : rsup(c,r,d) \wedge sub(d,d') \wedge rsub(d',r,e) \Rightarrow sub(c,e)$
($F_7$) $\forall c,r,d,s : rsup(c,r,d) \wedge psub(r,s) \wedge \Rightarrow rsup(c,s,d)$
($F_8$) $\forall c,r_1,r_2,r_3,d,e : rsup(c,r_1,d) \wedge rsup(d,r_2,e) \wedge pcom(r_1,r_2,r_3) \Rightarrow rsup(c,r_3,e)$
($F_9$) $\forall c : \neg sub(c,\bot)$

○

These definitions provide as with complete formalism to compute the most probable, coherent, and fully classified ontology from a probabilistic $\mathcal{EL}^{++}$-LL knowledge base, by calculating the MAP state of the corresponding Markov logic network.

### 2.5.2. Tuple-independent OWL

A TIP-OWL (tuple-independent OWL) knowledge base $\mathcal{TKB} = \langle \mathcal{T}, \mathcal{A}, P\rangle$ consists of a *DL-Lite*$_R$ TBox $\mathcal{T}$, an ABox $\mathcal{A}$, and a probability distribution $P : \mathcal{A} \to [0,1]$. Abusing terminology, we say that $\mathcal{KB} \subseteq \mathcal{TKB}$ if they have the same TBox and the ABox of $\mathcal{KB}$ is a subset of the ABox of $\mathcal{TKB}$. We adopt the independent tuple semantics in the spirit of the probabilistic semantics for logic programs proposed by De Raedt et al. (De

Raedt et al., 2007) (which is in turn based on distribution semantics (Sato, 1995) and answer set semantics (Poole, 1997)) and define the probabilistic semantics of a TIP-OWL knowledge base in terms of a distribution over possible knowledge bases as follows:

**Definition 2.9** (Tuple-independent OWL (TIP-OWL))
*Let $\mathcal{TKB} = \langle \mathcal{T}, \mathcal{A}, P \rangle$ be a TIP-OWL knowledge base. Then the probability of a DL-Lite$_R$ Knowledge Base $\mathcal{KB} = \langle \mathcal{T}, \mathcal{A}' \rangle \subseteq \mathcal{TKB}$ is given by:*

$$P(\mathcal{KB}|\mathcal{TKB}) = \prod_{a' \in \mathcal{A}'} P(a') \cdot \prod_{a \in \mathcal{A} \setminus \mathcal{A}'} (1 - P(a))$$

○

Based on this semantics, we can now define the probability of existential queries over TIP-OWL knowledge bases as follows. First, the probability of a query over a possible knowledge base is one if the query follows from the knowledge base and zero otherwise:

$$P(Q|\mathcal{KB}) = \left\{ \begin{array}{cc} 1 & \mathcal{KB} \models Q \\ 0 & \text{otherwise} \end{array} \right\}$$

Taking the probability of possible knowledge bases into account, the probability of an existential query over a possible knowledge base becomes the product of the probability of that possible knowledge base and the probability of the query given that knowledge base. By summing up these probabilities over all possible knowledge bases, we get the following probability for existential queries over TIP-OWL knowledge bases:

$$P(Q|\mathcal{TBK}) = \sum_{\mathcal{KB} \subseteq \mathcal{TKB}} P(Q|\mathcal{KB}) \cdot P(\mathcal{KB}|\mathcal{TKB})$$

This defines a complete probabilistic semantics for TIP-OWL knowledge bases and queries.

# Part II.

# Research Contributions

# 3 Debugging Large-scale Uncertain Temporal Knowledge Graphs

## 3.1. Introduction

The automated construction of knowledge graphs is an active area of research (Cafarella et al., 2011). Especially, Open Information Extraction (OIE) is used for extracting information from large amounts of web sites (Banko et al., 2007; Etzioni et al., 2008; Pujara et al., 2013) and constructing knowledge graphs such as YAGO (Hoffart et al., 2013), Google Knowledge Vault (Dong et al., 2014b), Freebase (Bollacker et al., 2008), ProbBase (Wu et al., 2012), ProbKB (Chen and Wang, 2014), and ReVerb (Fader et al., 2011). Additionally, there are also knowledge graphs curated mostly manually, like DBpedia (Lehmann et al., 2015) or Wikidata (Lehmann et al., 2015). Some of these knowledge graphs, like YAGO or Wikidata, also contain temporal facts – facts with validity time. Moreover, most of them store probabilistic facts, i.e., facts together with confidence scores witnessing how likely a facts is to hold.

Automated construction often produces noisy and inaccurate facts whose errors can then propagate during inference and knowledge expansion. This poses some key challenges: The first challenge is *providing temporal information* at all. Most existing approaches focus on identifying static facts encoded as binary relations. However, the vast majority of facts are fluents (dynamic relations whose truth is a function of time), only holding at a point in or during an interval of time. Facts like *Claudio Ranieri being coach of Chelsea F.C.* loose relevance without a temporal scope (*2000–2004* in this case).

Second, knowledge graphs have to be cleaned from inaccurate facts to reduce maintenance costs and improve the reliability of the contained information, and to allow for sound logical reasoning. Unfortunately, existing methods (e.g. as proposed by Schlobach et al. (2007)) are limited in their capability to handle facts that are both uncertain and temporal. This can result in situations where statements about an instance appear as inconsistent although they are valid if temporal information is considered and they related to different points in time. For example assume we have two statements about *Claudio Ranieri being coach of F.C. Valencia from 1997 to 1999* and *Claudio Ranieri being coach of Chelsea from 2000 to 2004*, and the constraint that a person can only be a coach for one team at a time. If temporal information is ignored, one of those statements has to be false for a knowledge base to be consistent. However, by considering the temporal information about the years those statements are relating to, we can assert that both statements are valid, as they are defined for different, not overlapping intervals in time.

Furthermore, little has been done in terms of techniques to debug uncertain knowledge graphs, with the exception of preliminary results in (Chekol et al., 2016; Chen and Wang, 2014; Dylla et al., 2011; Huber et al., 2014). In addition, temporal inference rules and constraints for consistency checking are not only useful for identifying conflicting facts, but also for deriving implicit facts from existing ones.

In this thesis we focus on the second challenge: Employing a formalism to define a set of temporal inference rules and constraints and identify the most probable and error-free temporal knowledge graph from an existing uncertain temporal knowledge graph. In particular, we developed a tool with an intuitive web interface for creating temporal constraints and explore the resulting knowledge graph.

## 3.2. Preliminaries

In this section we first introduce uncertain temporal knowledge graphs, an extension of regular uncertain knowledge graphs (cf. 2.1.1) with a temporal dimension. Furthermore, we describe a numerical extension for Markov logic networks and probabilistic soft logic, both used for temporal reasoning.

### 3.2.1. Temporal Knowledge Graphs

Temporal knowledge graphs are obtained by adding a temporal domain to regular knowledge graphs. As shown by Gutierrez et al. (2005) and Motik (2012) RDF knowledge graphs can be extended to temporal knowledge graphs by annotating the triples with a temporal element. Temporal databases usually distinguish between *valid time* and *transaction time* information (Tansel et al., 1993). *Valid time* describes at (or during) which point(s) in time a statement is true in the real world. *Transaction time* indicates when the information was known and stored in the database. In this work, we focus only on *valid time.*

Thus, we define a temporal knowledge graph as a knowledge graph whose statements are annotated with a closed interval for their valid time in the following way:

**Definition 3.1** (Temporal Knowledge Graph)
*Let $\mathcal{K}$ be knowledge graph as defined above, and $\mathcal{T}$ be a finite set of discrete points in time. A temporal knowledge graph $\mathcal{K}_\mathcal{T}$ is then the set $\{(s, p, o, [t_1, t_2]) : (s, p, o) \in \mathcal{K}, t_1, t_2 \in \mathcal{T}, t_1 \leq t_2\}$.* ○

Note that a single point in time (e.g. a date of birth) is easily modeled with an interval of length 0, i.e., $t_1 = t_2$.

### 3.2.2. Uncertain Temporal Knowledge Graphs

Uncertain temporal knowledge graphs are the combination of knowledge graphs with temporal *and* uncertain elements:

**Definition 3.2** (Uncertain Temporal Knowledge Graphs)
*Let $\mathcal{K}$ be knowledge graph as defined above, $\mathcal{T}$ be a finite set of discrete points in time and $\mathcal{W}$ be a set of elements expressing degrees of uncertainty. A uncertain temporal knowledge graph $\mathcal{K}_{\mathcal{T},\mathcal{W}}$ is then the set $\{(s,p,o,[t_1,t_2],w) : (s,p,o) \in \mathcal{K}, t_1, t_2 \in \mathcal{T}, t_1 \leq t_2, w \in \mathcal{W}\}$.* ○

Next, we briefly explain probabilistic soft logic which we use to calculate fast approximations of the exact results we otherwise compute with Markov logic networks.

### 3.2.3. Probabilistic Soft Logic

Probabilistic soft logic (PSL) combines graphical models with rules in a first-order syntax. Unlike MLNs, which are based on regular Markov networks, PSL is based on *hinge-loss* Markov networks (Bach et al., 2017). This allows for soft truth values in $[0,1]$ as opposed to the coarse boolean values in MLNs. The logical rules in PSL are thus interpreted using *Lukasiewicz logic* (Klir and Yuan, 1995) instead of classical boolean logic. The truth value of a formula $A$ is also called valuation and denoted by $v(A)$. PSL is only the notions of strong conjunction $\wedge$ with $v(A \wedge B) = max\{0, v(A) + v(B) - 1\}$ and strong disjunction $\vee$ with $v(A \vee B) = min\{1, v(A) + v(B)\}$. We show on a small example how Lukasiewicz logic is evaluated:

$$\begin{aligned}
&v(\text{friends}(Alice, Bob)) = 0.9 \qquad v(\text{hasHobby}(Alice, Football) = 0.8\\
&v(\text{friends}(Alice, Bob) \wedge \text{hasHobby}(Alice, Football))\\
&\quad = max\{0, v(\text{friends}(Alice, Bob)) + v(\text{hasHobby}(Alice, Football)) - 1\}\\
&\quad = max\{0, 0.9 + 0.8 - 1\}\\
&\quad = 0.7
\end{aligned} \tag{3.1}$$

The probability distribution over possible worlds in PSL is similarly defined to that for MLNs in Equation (2.3):

$$P(X = x) = \frac{1}{Z} \exp\left(-\sum_i w_i d(x_i)\right) \tag{3.2}$$

where Z is again a normalization constant, $w_i$ the weight of the $i$'th formula in world $x$, $d(x_i)$ the *distance to satisfaction* of formula $i$, and $p \in \{1,2\}$ a distance exponent. The distance to satisfaction is defined as $max\{0, v(x_{i,body}) - v(x_{i,head})\}$.

The PSL allows to configure the *activation threshold*, i.e. the minimum truth value an atom needs to have in order to be considered during inference. The default value is 0.01.

Probabilistic soft logic has two main limitations:

1. Weights need to be positive.
2. Rules are limited to conjunctive bodies.

However, at least the first point can for simple formulas be circumvented by using negation. For example, $\langle \text{friends}(x, y), -0.1\rangle$ can be rewritten as $\langle \neg\text{friends}(x, y), 0.1\rangle$.

### Numerical Extension

Markov logic networks and probabilistic soft logic can both been extended with numerical constraints (Chekol et al., 2017a, 2016)[1]. We denote the formalism of Markov logic networks extended with numeric constraints as $\text{MLN}_{NC}$. A formula $F_{NC}$ in an $\text{MLN}_{NC}$ consists of a regular MLN formula $F$ and a numerical constraint $\Phi$. $\Phi$ is built from arithmetic expressions (e.g. $+, \div, \surd, \ldots$) over variables in $F$ and numeric constants which are connected via comparison operators (e.g. $<, =, \neq, >, \ldots$). Comparison expressions can in turn be connected with boolean operators (e.g. $\vee, \wedge, \ldots$). An $\text{MLN}_{NC}$ is thus a set of pairs $(F_{NC,i}, w_i)$ with $F_{NC,i}$ being an FOL formula which may contain a numeric constraint, and $w_i$ the real-valued weight of the formula. The semantics of an $\text{MLN}_{NC}$ stays identical to that of regular MLNs.

The following preliminary example shows how numerical constraints can be used to define rules over temporal information.

**Example 3.1** (MLN with Numerical Constraints)
*We illustrate a numerical constraint over temporal information on the following formula:*

$$\begin{aligned}\langle &\text{friends}(x, y, [t_1, t_2]) \wedge \text{born}(x, [b_1, b_1]) \wedge \text{born}(y, [b_2, b_2]) \rightarrow \phi(t_1, t_2, b_1, b_2),\\ &\phi(t_1, t_2, b_1, b_2) = (t_1 < t_2) \wedge (b_1 < t_1) \wedge (b_2 < t_1), +\infty\rangle\end{aligned} \tag{3.3}$$

*The formula enforces that if two persons are* friends*, the interval during which they are friends needs to be valid (i.e. of non-zero length and start before it ends) and start after both of them are* born*.*

*Another example demonstrating arithmetic expressions within numerical constraints is the following formula checking that if a person's date of death and date of birth are known, its age is the difference between the two:*

$$\begin{aligned}\langle &\text{born}(x, [t_1, t_1]) \wedge \text{died}(x, [t_2, t_2]) \rightarrow \text{age}(x, c_1) \wedge \phi(t_1, t_2), \qquad (3.4)\\ &\phi(t_1, t_2) = (t_1 < t_2) \wedge c_1 = t_2 - t_1, +\infty\rangle \qquad \triangle\end{aligned}$$

We define the different types of numerical constraints more formally in Section 3.3.1.

[1]Numerical constraints should not be confused with arithmetic rules in PSL which fulfill a different task

### 3.2.4. Reasoning in Uncertain Temporal Knowledge Graphs

We will now formally define uncertain temporal knowledge graphs UTKG as extensions of temporal knowledge graphs and probabilistic graphical models that are capable of representing uncertainties and reasoning over temporal knowledge bases.

**Definition 3.3** (Uncertain Temporal Knowledge Graph (UTKG))
*Let $\mathcal{D} = \{f_i\}$ be a set of deterministic temporal facts and $\mathcal{U} = \{f_j, w_j\}$ be a set of temporal facts associated with a real-valued weight, with $\mathcal{D} \cap \mathcal{U} = \emptyset$. Then $\mathcal{G} = (\mathcal{D},\mathcal{U})$ is an uncertain temporal knowledge graph.* ○

We extend the membership ($\in$) and subset relations ($\subseteq$) for the use with UTKGs as follows:

**Definition 3.4** (Membership and Subset Relation in UTKGs)
*Let $\mathcal{G}$ and $\mathcal{G}'$ be two UTKGs and $f = (s, p, o, [t_1, t_2])$. We say that $f \in \mathcal{G}$ if $\exists(s, p, o, [t'_1, t'_2]) \in \mathcal{G}$ with $t'_1 \leq t_1$ and $t_2 \leq t'_2$. We also say that $\mathcal{G} \subseteq \mathcal{G}'$ iff $\forall f \in \mathcal{G} : f \in \mathcal{G}'$.* ○

With those definitions we can now define the semantics of uncertain temporal knowledge graphs. It is based on a joint probability distribution over the uncertain part $\mathcal{U}$ of the graph. As in MLNs, the weights of the facts define a log-linear probability distribution:

**Definition 3.5** (Semantics of UTKGs)
*Given an UTKG $\mathcal{G} = (\mathcal{D},\mathcal{U})$ the probability $P$ of $\mathcal{G}$ is defined as follows:*

$$P(\mathcal{G}) = \begin{cases} \frac{1}{Z} \exp\left( \sum\limits_{(f_i,w_i)\in\mathcal{U}:\mathcal{G}\models_t f_i} w_i \right) & \text{if } \mathcal{G} \models_t \mathcal{D} \\ 0 & \text{otherwise} \end{cases} \tag{3.5}$$

*where $Z$ is a normalization constant and $\models_t$ is a temporal entailment relation.* ○

Next, we define temporal inference rules, and introduce Herbrand models as a means to translate uncertain temporal knowledge graphs to first-order logic, and ultimately conduct reasoning in UTKGs.

**Definition 3.6** (Temporal Inference Rules)
*We denote $\mathcal{F}$ the following set of temporal inference rules $r_1 - r_8$ over an RDF/S*

*vocabulary:*

$$
\begin{array}{ll}
(r_1) & \mathtt{q}(a, \mathtt{isa}, \mathtt{Class}, T_1) \rightarrow q(a, \mathtt{sc}, a, T_1) \\
(r_2) & \mathtt{q}(a, \mathtt{sc}, b, T_1) \wedge q(b, \mathtt{sc}, c, T_2) \wedge \mathit{check}(T_1, T_2) \rightarrow q(a, \mathtt{sc}, c, T_3) \\
(r_3) & \mathtt{q}(a, \mathtt{sc}, b, T_1) \wedge q(x, \mathtt{isa}, a, T_2) \wedge \mathit{check}(T_1, T_2) \rightarrow q(x, \mathtt{isa}, b, T_3) \\
(r_4) & \mathtt{q}(a, \mathtt{isa}, \mathtt{Property}, T_1) \rightarrow q(a, \mathtt{sp}, a, T_1) \\
(r_5) & \mathtt{q}(a, \mathtt{sp}, b, T_1) \wedge q(b, \mathtt{sp}, c, T_2) \wedge \mathit{check}(T_1, T_2) \rightarrow q(a, \mathtt{sp}, c, T_3) \\
(r_6) & \mathtt{q}(a, \mathtt{sp}, b, T_1) \wedge q(x, a, y, T_2) \wedge \mathit{check}(T_1, T_2) \rightarrow q(x, b, y, T_3) \\
(r_7) & \mathtt{q}(a, \mathtt{dom}, c, T_1) \wedge q(x, a, y, T_2) \wedge \mathit{check}(T_1, T_2) \rightarrow q(x, \mathtt{isa}, c, T_3) \\
(r_8) & \mathtt{q}(a, \mathtt{ran}, d, T_1) \wedge q(x, a, y, T_2) \wedge \mathit{check}(T_1, T_2) \rightarrow q(y, \mathtt{isa}, d, T_3)
\end{array}
$$

$$
T_3 = [t_1, t'_1] \bowtie [t_2, t'_2] = \begin{cases} [t_1, t'_1] & \text{if } t_1 = t_2 \wedge t'_1 = t'_2 \\ [t'_1, t_2] & \text{if } t'_1 = t_2 \\ [t_2, t'_1] & \text{if } t_1 < t_2 \wedge t_2 < t'_1 \wedge t'_1 < t'_2 \\ [t_1, t'_1] & \text{if } t_1 < t_2 \wedge t'_1 < t'_2 \\ [t_1, t'_1] & \text{if } t_1 = t_2 \wedge t'_1 < t'_2 \\ [t_2, t'_2] & \text{if } t'_1 < t_1 \wedge t_2 = t'_2 \\ \emptyset & \text{if } t'_1 < t_2 \end{cases} \tag{3.6}
$$

$$
\mathit{check}(T_1, T_2) = \begin{cases} \mathit{false} & \text{if } T_1 \bowtie T_2 = \emptyset \\ \mathit{true} & \text{otherwise} \end{cases}
$$

*We abbreviate RDF/S vocabulary names as follows:* `isa` *for* `rdf:type`, `Class` *for* `rdfs:Class`, `sc` *for* `rdfs:subClassOf`, `Property` *for* `rdf:Property`, `dom` *for* `rdfs:domain`, `ran` *for* `rdfs:range`, *and* `sp` *for* `rdfs:subPropertyOf`. *The temporal interval* $[t_i, t'_i]$ *is denoted as* $T_i$. `q` *is an RDF quad. All of the formulas are universally quantified over all the variables.* ○

**Herbrand Models** Herbrand models are models for logical formulas which simply map all terms to themselves and predicates to relations over the universe of terms (the *Herbrand universe*). For the set of all constants $\mathcal{C}$ in an UTKG $\mathcal{G}$, the *Herbrand base* (additionally denoting relation symbols on top of the Herbrand universe) of $\mathcal{F}$ from Definition 3.6 can be constructed by instantiating all the variables in $\mathcal{F}$ using the constants in $\mathcal{C}$. Given the finite sets of constants $\mathcal{C}$ and time points $\mathcal{T}$, the function $\theta$ maps each fact of some UTKG into a subset of the Herbrand base *HB* of $\mathcal{F}$ with respect to $\mathcal{C}$ and $\mathcal{T}$. Each subset of the Herbrand base is a *Herbrand interpretation* specifying which ground atoms are true. A Herbrand interpretation $H$ is a Herbrand model of $\mathcal{F}$, denoted as $\models_H \mathcal{F}$, iff it satisfies all groundings of the formulas in $\mathcal{F}$.

**Definition 3.7** (Mapping UTKGs into FOL)
*Given a UTKG $\mathcal{G}$ over a finite set of constants $\mathcal{C}$, a time domain $\mathcal{T}$, and HB the Herbrand*

*base of $\mathcal{F}$ with respect to $\mathcal{C}$ and $\mathcal{T}$, then $\theta : \mathcal{P}(\mathcal{G}) \rightarrow \mathcal{P}(HB)$ maps $\mathcal{G}$ into subsets of HB as follows:*

$$\theta(\mathcal{G}) = \bigcup_{\mathtt{f} \in \mathcal{G}} \theta(\mathtt{f}), \text{ where } \theta\left((s, p, o, T)\right) = \mathtt{quad}(s, p, o, T)$$

$\bigcirc$

At this point, we need to establish that the function $\theta$ is bijective, i.e., it induces a one-to-one correspondence between the Herbrand models of $\mathcal{F}$ and expanded knowledge graphs. Applying $\mathcal{F}$ repeatedly on an uncertain temporal knowledge graph may generate a set of new facts. This results in an *expanded* knowledge graph.

**Theorem 3.1**
*Let $\mathcal{C}$ be a set of constants, $\mathcal{T}$ be a set of time points, $\mathcal{G}$ be an UTKG over $\mathcal{C}$, and HB be the Herbrand base of $\mathcal{F}$ with respect to $\mathcal{C}$. Then, for any $\mathcal{G}' \subseteq \mathcal{G}$, $\mathcal{G} \vDash_t \mathcal{G}' \Rightarrow \theta(\mathcal{G}') \vDash_H \mathcal{F}$ and for any $H \subseteq HB$, $H \vDash_H \mathcal{F} \Rightarrow \theta^{-1}(H) \vDash \mathcal{G}''$ and $\mathcal{G} \vDash_t \mathcal{G}''$.*

Figure 3.1 illustrates the different components of a Markov Logic Network, the grounding step, and the model checking step assigning truth values. The *predicates R* instantiated with *constants C* define all possible *random variables x*. In turn, all possible combinations of truth values for this variables result in the *possible worlds X*. The *formulas* are instantiated with the random variables, which then give as the *grounded formulas G*. Each combination of a possible world and the grounded formulas is then model checked, essentially determining their truth value and resulting in a cumulative weight for that world.

Relying on the above theorem, we can introduce MAP inference in uncertain temporal knowledge graphs.

### MAP Inference

*Maximum a posteriori* (MAP) inference in uncertain temporal knowledge graphs corresponds to obtaining the most probable, consistent knowledge graph. In simple words, when computing the MAP state we look at all possible worlds; for those which are consistent (i.e. no hard rules are violated) we calculate the sum of the weights of all soft formulas and facts. That world which has the highest sum – and thus is the most likely one – is called the MAP state.

More formally, given an UTKG $\mathcal{G}$, a set of inference rules $\mathcal{F}$, and a translation function $\theta$, we denote the MAP problem as $map(\theta(\mathcal{G}), \mathcal{F})$. When computing $map(\theta(\mathcal{G}), \mathcal{F})$, first, $\mathcal{G}$ is translated by $\theta$ to an equivalent Markov logic formalization. Then, the inference rules $\mathcal{F}$ are added to this translation.

The MAP state is computed with the help of a cutting planes algorithm (Chekol et al., 2016) applied to this input data. To do so, the evidence clauses $\theta(\mathcal{G})$ and the grounding of

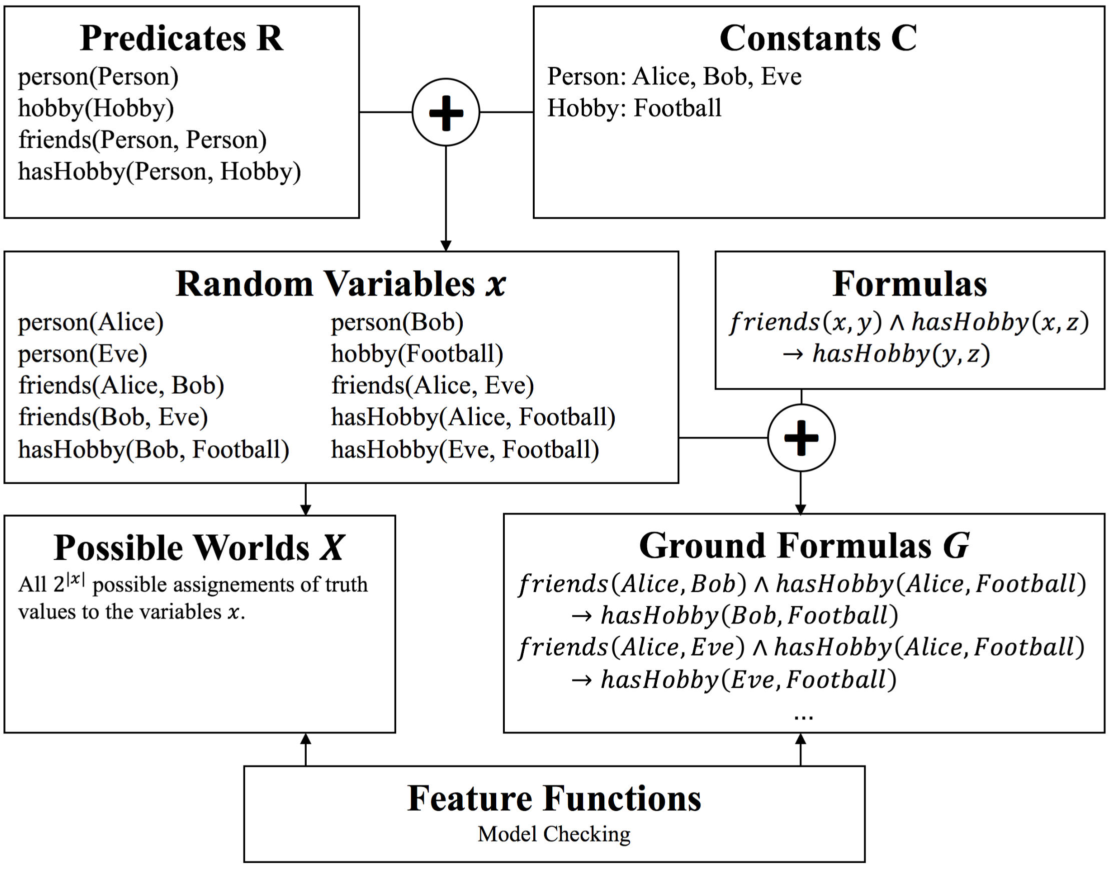


Figure 3.1.: The diagram describes the grounding of a Markov Network. The grounded formulas G are generated by substituting each occurrence of every variable in the MLN Formulas with constants of the domain C. The possible worlds X are generated by giving all possible groundings of each predicate. Both the possible worlds X and the grounded formulas G are checked and assigned a value of 1 if they are true, and 0 otherwise. (Adapted from (Jain, 2011; von Stülpnagel et al., 2014)).

$\mathcal{F}$ with respect to $\theta(\mathcal{G})$ are given as input. Applying the inverse translation function $\theta^{-1}$ to the MAP state yields the most probable temporal knowledge graph. The MAP problem in MLN can be turned into an integer linear program (Noessner et al., 2013), which allows to integrate external functions (e.g., to check the conditions in Definition 3.6).

**Theorem 3.2**
*Given the following:*

- *An UTKG $\mathcal{G} = (\mathcal{D},\mathcal{U})$ over a finite set $\mathcal{C}$ of constants, and a finite set of time points $\mathcal{T}$*
- *The Herbrand base HB of the formulas $\mathcal{F}$ with respect to $\mathcal{C}$ and $\mathcal{T}$*
- *The set of ground formulas $\mathcal{G}_1$ constructed from $\mathcal{D}$*
- *The set of ground formulas $\mathcal{G}_2$ constructed from $\mathcal{U}$*

*the most* probable, expanded *and* consistent *temporal knowledge graph is obtained with:*

$$\theta^{-1}(H) = \underset{HB \supseteq H \models \mathcal{G}_1 \cup \mathcal{F}}{\arg\max} \left( \sum_{(\mathtt{f}, w_j) \in \mathcal{G}_2 : H \models_{H} \mathtt{f}_j} w_j \right)$$

From Theorem 3.2 and the results in Chekol et al. (2016) it follows that the problem of computing the most probable temporal knowledge graph is NP-hard.

**Example 3.2** (MAP State)
*We recall the predicates and rules from Equation* (2.4) *and Equation* (3.3)*:*

$$\begin{gathered}\{\ \text{person}(\mathit{Person}), \text{friends}(\mathit{Person}, \mathit{Person}, [\mathcal{T}, \mathcal{T}]), \\ \text{hobby}(\mathit{Hobby}), \text{hasHobby}(\mathit{Person}, \mathit{Hobby}), \text{born}(\mathit{Person}, [\mathcal{T}, \mathcal{T}])\ \}\end{gathered} \tag{3.7}$$

$$\langle \text{friends}(x, y, [t, t']) \wedge \text{hasHobby}(x, z) \rightarrow \text{hasHobby}(y, z), 2\rangle \tag{3.8}$$

$$\begin{aligned}\langle \text{friends}(x, y, [t, t']) \wedge \text{born}(x, [b_1, b_1]) \wedge \text{born}(y, [b_2, b_2]) \rightarrow \phi(t, t', b_1, b_2), \\ \phi(t, t', b_1, b_2) = (t \leq t') \wedge (b_1 \leq t) \wedge (b_2 \leq t), +\infty\rangle\end{aligned} \tag{3.9}$$

$$\begin{gathered}\langle \text{friends}(x, y), -0.1\rangle \\ \langle \text{hasHobby}(x, y), -0.1\rangle\end{gathered} \tag{3.10}$$

*and the evidence given in Equation* (2.5) *extended with temporal information and (uncertain) birthdates for each person:*

$$\begin{matrix}\langle \text{person}(\mathit{Alice}), +\infty\rangle & \langle \text{person}(\mathit{Bob}), +\infty\rangle \\ \langle \text{person}(\mathit{Eve}), +\infty\rangle & \langle \text{hobby}(\mathit{Football}), +\infty\rangle\end{matrix} \tag{3.11}$$

$$\langle \text{friends}(\mathit{Alice}, \mathit{Bob}), [1988, \textsc{Now}], 3.5\rangle \tag{3.12}$$

$$\langle \text{hasHobby}(\mathit{Alice}, \mathit{Football}), 1.8\rangle \tag{3.13}$$

$$\langle \text{born}(\mathit{Alice}, [1990, 1990]), 1\rangle \tag{3.14}$$

$$\langle \text{born}(\mathit{Bob}, [1984, 1984]), 1.7\rangle \tag{3.15}$$

*Two consistent possible worlds and with the overall highest sum of weights are shown below:*

$$\left.\begin{array}{c}\text{friends}(\mathit{Alice}, \mathit{Bob}), [1988, \textsc{Now}]\\ \text{hasHobby}(\mathit{Alice}, \mathit{Football})\\ \text{born}(\mathit{Bob}, [1984, 1984])\end{array}\right\} \qquad \sum w_i = \quad 6.3$$

---

$$\left.\begin{array}{c}\text{hasHobby}(\mathit{Alice}, \mathit{Football})\\ \text{born}(\mathit{Alice}, [1990, 1990])\\ \text{born}(\mathit{Bob}, [1984, 1984])\end{array}\right\} \qquad \sum w_i = \quad 4.5$$

*The other consistent possible worlds are subsets of those two or contain additional groundings for the predicates* friends *or* hasHobby*, which are discouraged by Equation* (3.10)*. Subsequently, they will always have a lower sum of weights and will not be considered as MAP state.*

*The difference between the two shown worlds is that one contains Equation* (3.12)*, and the other Equation* (3.14)*. These two equations conflict due to Equation* (3.9)*. As Equation* (3.12) *has the higher weight, the other possible world containing Equation* (3.14) *is discarded, and Equation* (3.16) *is determined as MAP state.* △

## Conditional Probability Inference

Given an uncertain temporal knowledge graph $\mathcal{G}$, the conditional probability of a temporal fact $\texttt{f}$ is the sum of the probabilities of all possible temporal knowledge graphs that are a consistent subset of $\mathcal{G}$ containing $\texttt{f}$. In general, a *conditional probability query* is a conjunction of a set of temporal facts given some uncertain temporal knowledge graph. Given a query $q$ and an UTKG $\mathcal{G}$, the conditional probability of $q$ is given by:

$$P_q(q \mid \mathcal{G}) = \sum_{\mathcal{G}' : q \subseteq \mathcal{G}' \subseteq \mathcal{G}} P(\mathcal{G}') \tag{3.18}$$

where $\mathcal{G}'$ is a possible world over the same constants $\mathcal{C} \cup \mathcal{T}$ as $\mathcal{G}$. The computation of conditional probabilities is also called *marginal inference*.

In order to sum over all possible consistent $\mathcal{G}'$ we need to compare time intervals in the facts of $q$ with those of $\mathcal{G}$. If the valid time ranges of the temporal facts in $q$ do not appear in $\mathcal{G}$, we try to rewrite the query $q$ as follows: for each temporal fact $\texttt{f} \in q$ if $\exists \texttt{f}' \in \mathcal{G}$ such that $\texttt{f} \subseteq^{+} \texttt{f}'$, we replace $\texttt{f}$ in $q$ with $\texttt{f}'$. The relation $\subseteq^{+}$ is defined as follows: Given two temporal facts $\texttt{f} = (s, p, o, [t_1, t_1'])$ and $\texttt{f}' = (s, p, o, [t_2, t_2'])$, $\texttt{f} \subseteq^{+} \texttt{f}'$ if $t_2 \leq t_1$ and $t_1' \leq t_2'$. In other words, we look for a fact $\texttt{f}'$ such that $\texttt{f}$ is valid *during* the interval in which $\texttt{f}'$ is.

The rewriting can be done in polynomial time in the size of the uncertain temporal knowledge graph, in the worst case. For instance, given the knowledge graph $\mathcal{G}$ in Example 3.2, the conditional query $q((\text{friends}(\mathit{Alice}, \mathit{Bob}), [2001, 2003]) \mid \mathcal{G})$ is rewritten as: $P_q(q((\text{friends}(Alice, Bob), [1988, \textsc{Now}]) \mid \mathcal{G})$. Since no additional computation is required, the complexity of conditional probability inference remains #P-hard for uncertain temporal knowledge graphs.

As conditional inference is intractable, computing exact probabilities is hard. Thus, sampling is commonly used to approximate probabilities. State-of-the-art marginal inference algorithms, like MC-SAT, are based on Markov chain Monte Carlo (MCMC) and Gibbs sampling (Poon and Vanderwende, 2010; Poon et al., 2008; Singh et al., 2012). They sample from the consistent (or conflict-free) temporal knowledge graphs according to the distribution $P_q$. This is very difficult for three reasons:

1. The complexity of reasoning in MLN
2. The size of uncertain knowledge graphs (e.g. NELL, ReVerb)
3. The presence of deterministic dependencies in uncertain temporal knowledge graphs

Because of these reasons, lifted inference techniques are proposed to be used for marginal inference (Singla and Domingos, 2008; Venugopal and Gogate, 2012).

Having established the mechanisms for inference in uncertain temporal knowledge graph, we can now define how we can detect (temporal) conflicts in uncertain knowledge graphs.

## 3.3. Conflict Detection in Uncertain Temporal Knowledge Graphs

Uncertain knowledge graphs can contain a large number of numerical data like dates, times, latitudes/longitudes, or other numerical values measured in different units. Those facts can be conflicting, especially in automatically created knowledge graphs. One way of resolving such errors is to use a set of (probabilistic) constraints and compute a MAP state of a given knowledge graph, which effectively throws out facts that have inferior weights or confidences. However, this approach is often too coarse and simple. Consider for example the following knowledge graph about Alice's *height*, implicitly stated in meters:

$$\langle \textit{height}(Alice, \textit{1.76}), \textit{0.6}\,\rangle$$
$$\langle \textit{height}(Alice, \textit{5.8}), \textit{0.9}\,\rangle$$

Assume that these facts are translated into an MLN framework along with the constraint that the property *height* is functional, i.e., $height(x, y) \wedge height(x, y') \rightarrow y = y'$. Performing MAP inference on this MLN results in a knowledge graph containing only the fact $\textit{height}(Alice, \textit{5.8})$, as it has the higher weight of the two conflicting facts. If we expect only heights in meters, this is not the desired result, as humans are not taller than 3m[2]. In order to rule out such conflicts, we will now show how to define additional numerical constraints and use them for conflict detection.

### 3.3.1. Numerical Constraints for Conflict Detection

Constraints are extensively used in description logics and database systems to ensure data validity. In the following, we introduce constraints to ensure validity of numerical attributes in uncertain knowledge graphs. Moreover, the constraints will also serve to extend the schema of the underlying knowledge graph.

In Datalog, a constraint is an expression of the form *body* $\rightarrow$ *head*, where the *head* is an atom (i.e., an expression of the form $p(x_1, \ldots, x_n)$ in which each $x_i$ is either a constant or a variable) and *body* is a set of atoms, such that each variable occurring in the *head* also occurs in some atom in the *body* (Abiteboul and Vianu, 1991). Since our definition of MLNs with numerical constraints allows to use external functions whose truth values are computed outside the MLN setting, we can express Datalog constraints (specifically, *inclusion dependencies*, *equality generating dependencies* and *negative constraints* (Abiteboul and Vianu, 1991)) with numerical constraints.

To debug uncertain knowledge graphs we introduce a number of those Datalog-inspired constraints which are formulated as *hard* (deterministic) or *soft* (uncertain) formulas in the MLN. For example, a rule which forbids persons to be taller than 3m looks as follows:

$$person(x) \land height(x, y) \rightarrow y < 3 \tag{3.19}$$

We will now formally describe three different types of constraints based on Datalog constraints.

**Definition 3.8** (Inclusion Dependencies with Inequalities (IDIs))
*An inclusion dependency with inequalities is a first-order logic formula of the form*

$$\forall x, y : \Phi(x, y) \land \mathsf{NC}(x', y') \rightarrow \Psi(y) \tag{3.20}$$

*where $\Phi(x, y)$ is the body of the formula (a conjunction of atoms), $\mathsf{NC}(x', y')$ denotes a numerical constraint as given in Section 3.3.1, and $\Psi(y)$ is the head of the formula, with $x$ and $y$ being sets of variables, and $x' \subseteq x$ and $y' \subseteq y$.* ○

**Example 3.3** (Inclusion Dependencies with Inequality Constraints)
*A constrained defining football players above the age of 40 as retired looks as following:*

$$\begin{aligned} &Footballer(x) \land age(x, y) \land \mathsf{NC}(y) \\ &\quad \rightarrow RetiredFootballer(x), \mathsf{NC}(y) = y > 40 \end{aligned} \tag{3.21}$$

△

[2]The tallest person in history was 2.72m: `http://www.guinnessworldrecords.com/world-records/tallest-man-ever`

**Definition 3.9** ((In)equality Generating Dependencies (IGDs))
*(In)equality generating dependencies are first-order formulas of the form*

$$\forall x : \Phi(x) \rightarrow \mathsf{NC}(x') \tag{3.22}$$

*where $\Phi(x)$ is a conjunction of atoms and $x' \subseteq x$.* ○

**Example 3.4** ((In)equality Generating Dependency Constraints)
*Temperature values in Celsius $t_C$ can be converted into an equivalent value in Fahrenheit scale $t_F$ using the formula $t_F = 1.8t_C + 32$. Written as a constraint which enforces that temperatures denote the same absolute value if given in Celsius and Fahrenheit, this looks as follows:*

$$\begin{aligned} &\mathit{tempC}(x, t_C) \wedge \mathit{tempF}(x, t_F) \\ &\quad \rightarrow \mathsf{NC}(t_C, t_F), \mathsf{NC}(t_C, t_F) = t_F = 1.8t_C + 32 \end{aligned} \tag{3.23}$$

△

**Definition 3.10** (Disjointness Constraints (DCs))
*Disjointness constraints are first-order formulas of the form*

$$\forall x : \Phi(x) \wedge \mathsf{NC}(x_i) \rightarrow \bot. \tag{3.24}$$

○

**Example 3.5** (Disjointness Constraints)
*Using DCs we can for example formulate the constraint that the valid life span of a person is greater than 0 and less than 150 years as follows:*

$$\begin{aligned} &\mathit{born}(x, b) \wedge \mathit{died}(x, d) \wedge \mathsf{NC}(b, d) \rightarrow \bot, \\ &\quad \mathsf{NC}(b, d) = 0 < (d - b) \wedge (d - b) < 150 \end{aligned} \tag{3.25}$$

△

Once an uncertain knowledge graph is translated into an equivalent Markov logic formalism using the mapping function $\theta$, and sets of IDIs, IGDs, and DCs over the knowledge graph have been constructed, we can apply MAP inference in order to retrieve the most probable and conflict-free graph using $map(\theta(\mathcal{G}), \mathcal{F}, \mathcal{C})$.

## 3.4. Datasets of Temporal Knowledge Graphs

Currently, uncertain knowledge graphs with rich temporal information are scarce, especially regarding valid times for a majority of the statements. As two exemplary sources for knowledge graphs, we used The Football Database[3], and we generated a large uncertain temporal knowledge graph form Wikidata[4], which already contains a lot of valid time annotations for statements, but no confidence values. Furthermore, we experimented with rule mining in AMIE (Galárraga et al., 2015) to extract rules from YAGO, which contains some temporal information and uncertainty.

### 3.4.1. FootballDB

`footballdb.com` lists detailed data about National Football League (NFL) players, mostly in tabular form. This table data often contains numeric and temporal information. Recently, (Web) table data extraction has attracted considerable attention from the data mining community (cf. for example Ritze (2017); Ritze et al. (2016)). Inspired by this, we extracted temporal facts about NFL players, that contains two important temporal relations: *playsFor* and *birthdate*. We extracted >13K temporal facts for the *playsFor* relation and >6K facts for the *birthdate* relation.

### 3.4.2. Wikidata

Wikidata is an open source knowledge bases which contains structured data that is used in many other Wikimedia projects, most prominently Wikipedia. Information is mostly entered and edited manually, but APIs for automatic editing also exist. In 2017, Wikidata contains about 46 million facts, of which we extracted over 6.3 million temporal facts. We extracted temporal information (especially also *valid times*) for various relations including *playsFor* (>4 million facts), *memberOf* (>23K), *spouse* (>20K), *educatedAt* (>6K), *occupation* (>4.5K), ...

### 3.4.3. Mining Rules from YAGO

The extraction of temporal rules and constraints is a well-known problem and there exist sophisticated tools for automating this process, e.g. AMIE (Galárraga et al., 2015) as one of the most well-known. Using AMIE, we mined rules from the YAGO dataset, for example:

- A person's birth date is before his death date:
  $\langle \text{born}(x, t_1) \wedge \text{died}(x, t_2) \Rightarrow \text{before}(t_1, t_2), 0.968\rangle$
- Birth and death date are functional:
  $\langle \text{born}(x, t_1) \wedge \text{born}(x, t_2) \Rightarrow t_1 = t_2, 0.734\rangle$
  $\langle \text{died}(x, t_1) \wedge \text{died}(x, t_2) \Rightarrow t_1 = t_2, 0.686\rangle$

[3]`http://footballdb.com`
[4]`http://wikidata.org`

Those rules are rather simplistic, but still helpful to catch many common errors. However these rules are not sufficient to capture many more complex conflicts (e.g., valid life span of a person, or a footballer cannot play for two clubs at the same time). Thus, we handcrafted more complex constraints that are used to identify conflicts in uncertain temporal knowledge graphs. We used several of these constraints in order to detect conflicts in The Football Database and Wikidata datasets.

## 3.5. Tool Support

We developed an intuitive web interface which integrates with multiple solvers for probabilistic graphical models. It executes the whole process of uploading RDF data, translating it to the appropriate formalism, adding constraints and inference rules (either predefined or newly created in the user interface), running MAP inference, and parsing and presenting the resulting consistent data and conflicts. We named this tool TeCoRe – Temporal Conflict Resolution in knowledge graphs. TeCoRe is open source and available here: `https://github.com/dwslab/TeCoRe`.

### 3.5.1. System Overview

The system architecture of TeCoRe is shown in Figure 3.2, which depicts the main components of the tool: a Web-based user interface, the translator which transform the input data to the appropriate formalism and adds predefined and user-defined inference rules and constraints, and the different solvers for probabilistic graphical models from which the user can choose. The tool can either run locally in an embedded server (this

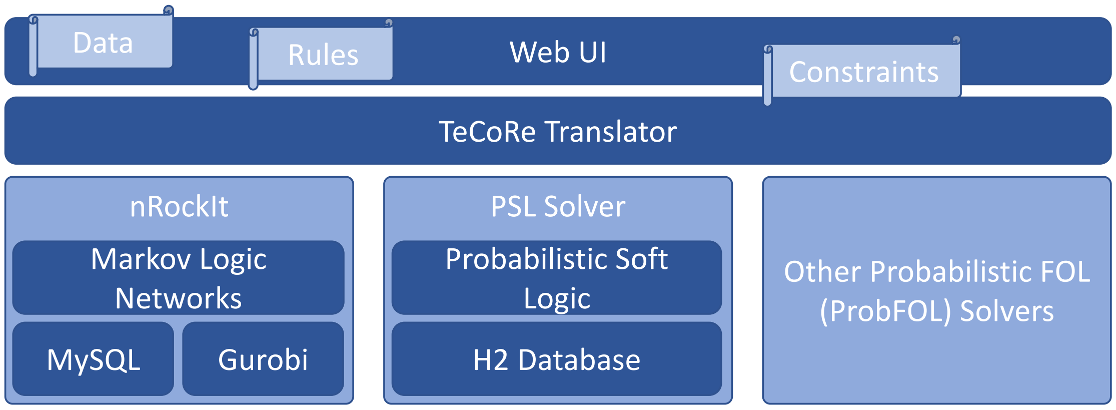


Figure 3.2.: TeCoRe system overview.

also requires solvers to be available locally), or be deployed on a dedicated server and accessed and used by multiple users simultaneously. In the following, we provide a brief description of the components.

### Web-based User Interface

The web UI is the main point of interaction for users of TeCoRe. Working with it involves four steps:

1. Choosing between nRockIt and PSL as solvers (or others)
2. Uploading a knowledge graph, inference rules and constraints
3. Adding additional temporal constraints based on Allen's interval algebra (see below)
4. Running inference to compute the most probable, conflict-free knowledge graph and inspecting and browsing consistent and inconsistent triples

For demonstration purposes, the user can choose from predefined datasets, for example the Football Database and Wikidata knowledge graphs described above including matching constraints. If the user wants to use his own data with custom inference rules

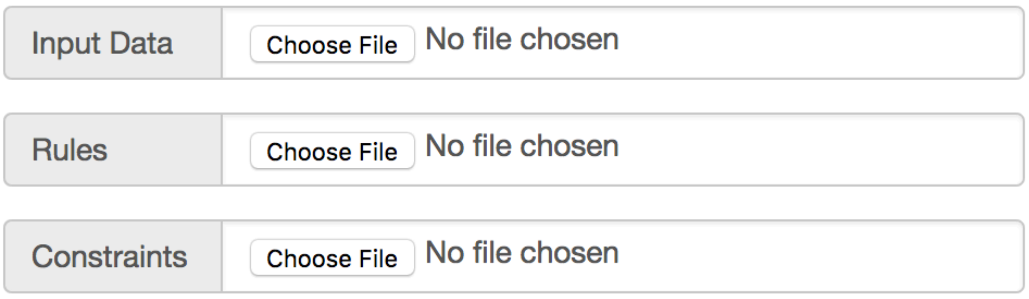
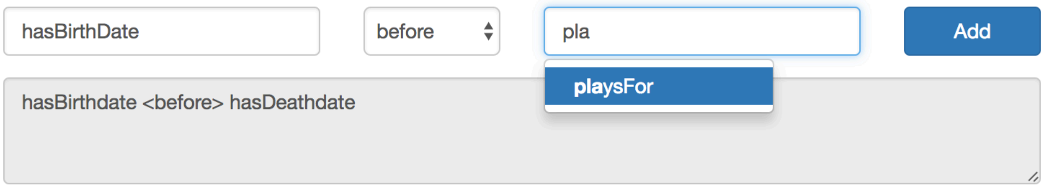


Figure 3.3.: Interface to select the input data, inference rules, and temporal constraints.

and constraints, the user interface also allows to upload those (Figure 3.3).

After uploading the data, the user is presented with a simple interface to add additional temporal constraints. This interface is intentionally limited to simple constraints formulated in Allen's interval algebra to be accessible to novice users not experienced with first-order logic constraints. The form offers two fields to enter predicates, supporting



hasBirthdate <before> hasDeathdate

Figure 3.4.: Constraints editor (predicate auto-completion).

the user with an auto-complete function, and a dropdown to choose the appropriate Allen relation (Figure 3.4.

**Allen's Interval Algebra** defines relations and an accompanying logic for temporal reasoning (Allen, 1983). The algebra defines 13 mutually exclusive relations which fully cover all possible temporal relationships between two intervals of time. Figure 3.5 shows

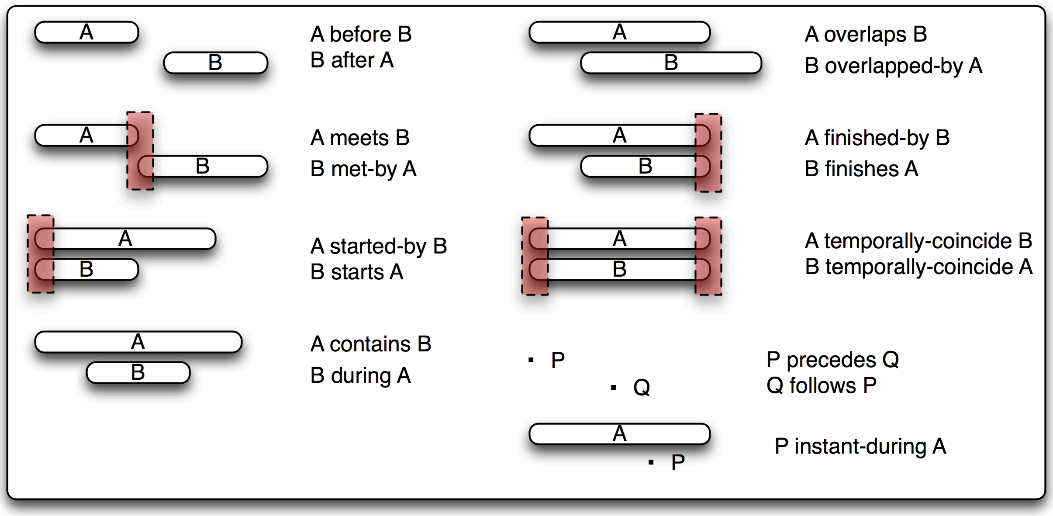


Figure 3.5.: Illustration of Allen's interval algebra (Jobczyk, 2016).

those relations (and two extension for points in time) and how two intervals A and B relate according to them.

Allen's interval algebra also describes logical rules to reason over multiple relations in the form of a composition table.

**Example 3.6** (Allen's Interval Algebra)
*To illustrate statements in Allen's Interval Algebra and possible inferences, consider the following two sentences:*

*After playing football, Alice has dinner.*

*During dinner, she watches the news.*

*Formalized in Allen's Interval Algebra they look as follows ({...}) indicates the possible relations):*

$$\begin{array}{rcl} \langle reading(Alice, Newspaper) & \{during,\ starts,\ finishes\} & eating(Alice, Dinner)\rangle \\ \langle going(Alice, Bed) & \{after,\ meets\} & eating(Alice, Dinner)\rangle \end{array}$$

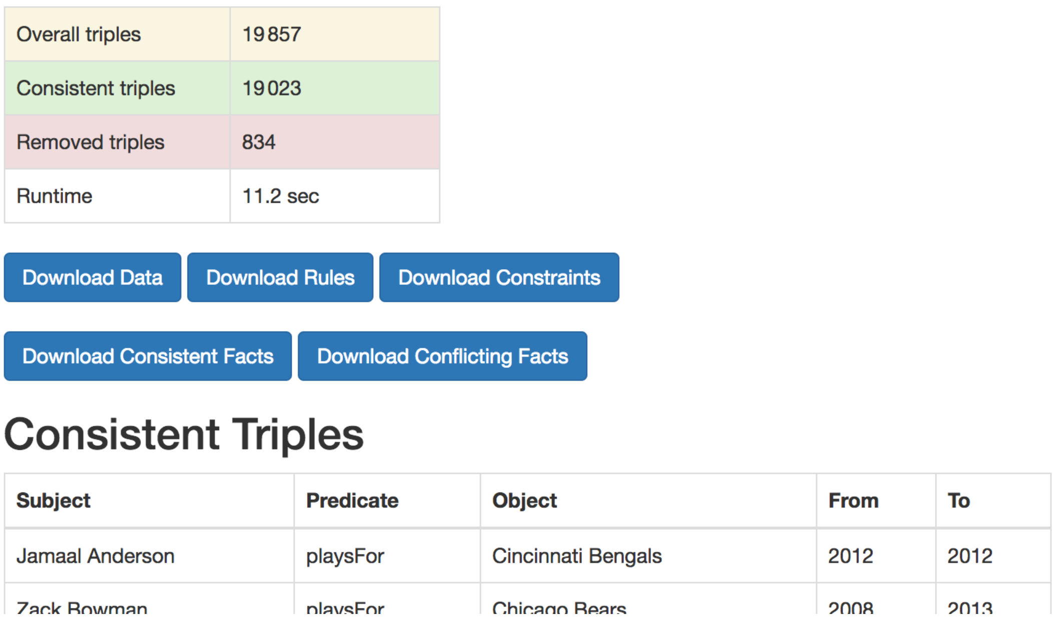
*From those two statements one can then infer that* $\langle going(Alice, Bed)\{after, meets\}\ reading(Alice, Newspaper)\rangle$ *holds.* △

### TeCoRe Translator and Solver

The translator runs after uploading the data and optionally defining additional temporal constraints. It parses data, inference rules, and temporal constraints, and transforms those into the specific syntax of the chosen solver (i.e. nRockIt or PSL). Special care is taken to verify that the input adheres to the expressivity of the solver. The explicit translation step allows to extend TeCoRe with additional solvers in the future, e.g. for Tractable MLN or Bayesian logic networks. The translator also automatically starts the solver and runs inference.

### Presentation of Results

The results page looks as depicted in Figure 3.6. The users is shown a summary statistics



| | |
|---|---|
| Overall triples | 19 857 |
| Consistent triples | 19 023 |
| Removed triples | 834 |
| Runtime | 11.2 sec |

Consistent Triples

| Subject | Predicate | Object | From | To |
|---|---|---|---|---|
| Jamaal Anderson | playsFor | Cincinnati Bengals | 2012 | 2012 |
| Zack Bowman | playsFor | Chicago Bears | 2008 | 2013 |

Figure 3.6.: Display of result statistics and result data (with browsable consistent and conflicting statements).

of the full number of parsed triples, the size of the conflict-free knowledge graph, and the number of removed, conflicting statements, as well as the runtime of the solver. Furthermore, the user can browse through a list of the consistent and the conflicting triples. All the data – uploaded by the user, and the computed consistent and conflicting statements – are offered for download for further investigation or processing.

## 3.6. Experiments

We ran two sets of experiments to evaluate the viability and usefulness of our approach:

1. Measurements of the overall runtime performance of MAP inference.
2. Performance analysis for the detection of temporal conflicts in uncertain temporal knowledge graphs, and how well the approach works with large amounts of noisy data.

Both sets of experiments were conducted over the Football Database and varying sizes of the Wikidata datasets, and with both solvers; nRockIt and PSL. We do not investigate individual errors found in the data.

We used the default configuration parameters for nRockIt and PSL. The runtimes were obtained on a virtual machine with 4 CPU cores and 16 GB of memory, and averaged over 10 runs.

### 3.6.1. Performance of MAP Inference

In the first experiment we ran both nRockIt and PSL on the Football Database (FDB) and Wikidata datasets described in Section 3.4. As the FDB dataset is small we simply used the whole dataset. For Wikidata, we generated random samples of increasing size to run the test on The results are listed in Table 3.1. From the runtimes on the different

| Dataset | FDB | Wikidata | | | | | | |
|---|---|---|---|---|---|---|---|---|
| (size) | 19k | 10k | 20k | 50k | 100k | 200k | 400k | 500k |
| nRockIt | 12.18 | 55.16 | 104.22 | 182.84 | 363.52 | 993.17 | *TO* | *TO* |
| PSL | 6.13 | 15.46 | 27.32 | 46.21 | 86.42 | 231.43 | 523.25 | 675.12 |

Table 3.1.: Runtime Performance (in seconds, averaged over 10 runs) over the Football Database (FBD) and various sizes of Wikidata for nRockIt and PSL. *TO* denotes a timeout after 30 minutes.

uncertain knowledge graphs, we can make two observations:

1. PSL performs between 2x–4.3x faster than nRockIt;
2. Performance for nRockIt and PSL on the FDB dataset is 8.6x and 4.5x better, respectively, than on a Wikidata graph of roughly equal size.

The first point shows that use of soft truth values – and thus simplifying the optimization problem when computing the MAP state while in turn sacrificing some accuracy (and expressiveness of constraints which, however, did not matter here) – does indeed significantly improve performance. Overall, both nRockIt and PSL scale linearly in the size of the Wikidata knowledge graph, as can also be seen in the graph of the runtimes shown in Figure 3.7.

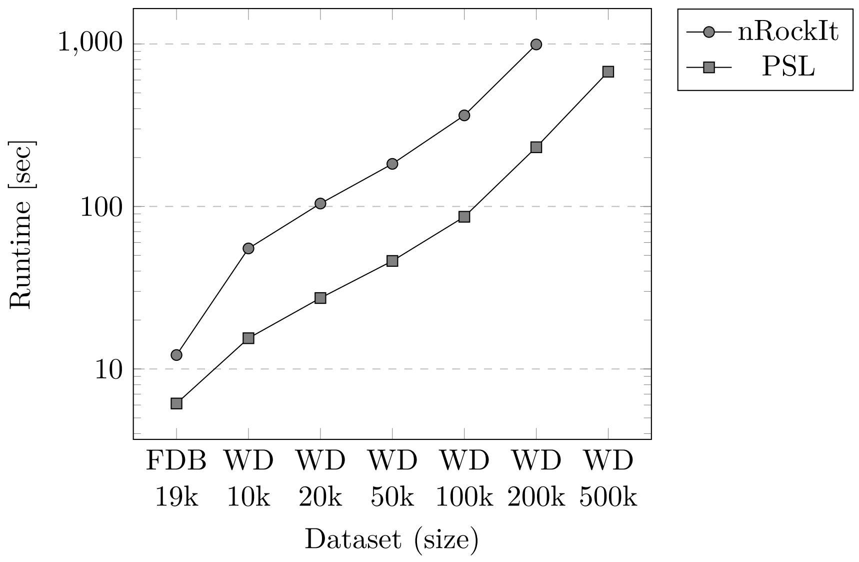


Figure 3.7.: Runtimes on datasets of various sizes

### 3.6.2. Performance of Conflict Detection

In the second set of experiments we focused on how well the approach works when the data gets more noisy. We used the conflict-free Wikidata temporal knowledge graph containing 50k statements, which we computed in the previous experiment, as a baseline. From this dataset with 0% conflicting statements we generate additional uncertain temporal knowledge graphs by inserting an additional 10%, 25%, 50%, 75%, and 100% of erroneous data, respectively.

We ran MAP inference on these and computed precision, recall and $F_1$ measure by comparing the resulting graph to the baseline. Precision (P) denotes the amount of correct statements retained from the baseline in relation to the overall size of the result. Recall (R) is the amount of correct triples in the result in relation to the size of the baseline. The $F_1$ measure is defined as follows:

$$F_1 = \frac{2PR}{P+R} \tag{3.26}$$

The numbers are shown in Table 3.2. From those results we can make two observations: 1) For both, nRockIt and PSL, precision is heavily impacted by noisy data. 2) Considering recall, nRockIt clearly outperforms PSL.

| Injection | | 0% | 10% | 25% | 50% | 75% | 100% |
|---|---|---|---|---|---|---|---|
| nRockIt | P | 1.00 | 0.91 | 0.83 | 0.67 | 0.55 | 0.50 |
| | R | 1.00 | 0.95 | 0.94 | 0.94 | 0.94 | 0.93 |
| | F1 | 1.00 | 0.93 | 0.88 | 0.78 | 0.69 | 0.65 |
| | Runtime | 182.84 | 263.83 | 344.36 | 392.05 | 415.42 | 665.26 |
| PSL | P | 1.00 | 0.93 | 0.85 | 0.70 | 0.63 | 0.53 |
| | R | 1.00 | 0.94 | 0.86 | 0.81 | 0.72 | 0.70 |
| | F1 | 1.00 | 0.93 | 0.85 | 0.75 | 0.67 | 0.60 |
| | Runtime | 46.21 | 52.63 | 55.82 | 69.02 | 82.24 | 93.62 |

Table 3.2.: Precision (P), recall (R), $F_1$ measure, and runtime (in seconds, averaged over 10 runs) for running the MAP inference with increasing percentage of wrong temporal facts injected. The Wikidata 50k dataset was used as baseline.

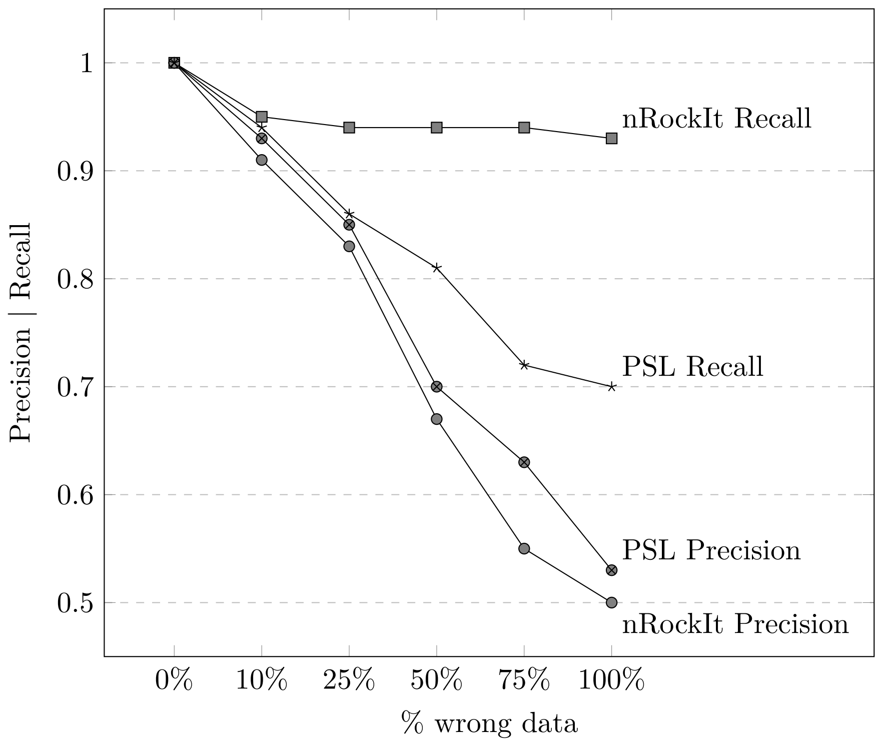


Figure 3.8.: Precision and recall for nRockIt and PSL on datasets inject with various amounts of wrong data.

The graph in Figure 3.8 more clearly shows the decrease in precision and recall for the two solvers.

We suppose that the reason nRockIt has the more *extreme* results for precision and recall – with PSL being somewhere in between – is caused by similar effects as mentioned in Section 1.1 and observed in Watson by Kalyanpur et al. (2012): In IBM Watson possible information could not be included when in doubt due to the rigid truth requirements of ontologies.

The same is true for MLNs, where only Boolean truth values are possible; however, in our case we have the opposite goal: instead of identifying and adding new information, we want to remove wrong information. Thus, when trying to remove wrong information, this is only done when we have a high degree of certainty, resulting in high recall and low precision, i.e. nRockIt tends to keep statements even if they are wrong instead of falsely removing true statements. For the same reason, as PSL allows for more uncertainty, its precision and recall values are somewhere in between nRockit's numbers: it sometimes removes wrong statements that nRockIt kept, but sometimes also true information retained by nRockIt.

However, if we look at the $F_1$ measure (Figure 3.9), we see that the overall performance of both solvers does not differ significantly, with only a slight advantage for nRockIt. There is rather a different trade-off between precision and recall for the two.

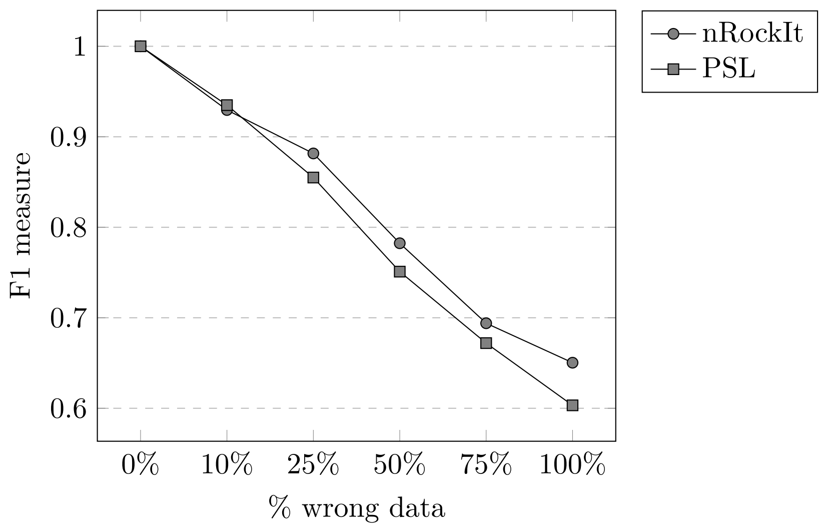


Figure 3.9.: F1 measure for nRockIt and PSL on datasets inject with various amounts of wrong data.

## 3.7. Related Work

Despite the general complexity of MLNs, it has been shown that it can be used to reason about facts extracted at Web scale using a combination of hand-crafted Schoenmackers et al. (2008) and extracted inference rules Schoenmackers et al. (2010). MLNs can be used to deal with temporal relations in open information extraction Ling and Weld (2010) or check the consistency of knowledge bases Chen and Wang (2014). The preliminary results that highlight the use of Markov Logic Networks to debug temporal knowledge graphs and on which this chapter is based are presented in Chekol et al. (2016); Huber et al. (2014). The basic idea in theses works is to use hand-crafted constraints to identify conflicts in knowledge bases containing date and time datatype values. However, this study

1. does not provide a formal characterization in terms of syntax and semantics,
2. only considers a subset of RDF(S) inference rules, and
3. does not consider constraints for debugging numerical attributes.

Dylla et al. (2013) and Papaioannou and Bohlen (2016) extend probabilistic databases with a temporal dimension. Furthermore, in earlier work Dylla et al. (2011) proposed an approach for resolving temporal conflicts in RDF knowledge bases. They used first-order logic Horn formulas with temporal predicates to express temporal and non-temporal constraints. However, these approaches are limited to a small set of temporal patterns and only allow for uncertainty in facts. Moreover, extending knowledge graphs using open domain information extraction will often also lead to uncertainty about the correctness of schema information. A large variety of temporal inference rules and constraints, some of which will be domain specific, can also be the subject of uncertainty. Chen and Wang (2014) debug erroneous facts by using a set of functional constraints, although they do not deal with numerical and temporal facts at the same time.

Knowledge base expansion and query-driven inference based on Markov Logic Networks have been studied in (Zhou et al., 2016). Contrary to TeCoRe, the knowledge bases considered are not temporal. Despite the general complexity of MLNs, it has been shown that this tool can be used to reason about facts extracted at Web scale using a combination of hand-crafted and extracted inference rules. MLNs can be used to deal with temporal relations in open information extraction (Ling and Weld, 2007) or check the consistency of knowledge bases (Chekol et al., 2017b).

We chose PSL over Tractable Markov Logic (TML) (Domingos and Webb, 2012) because it retains most of the rich expressiveness of MLN while being scalable. On the other hand, TML imposes heavy restrictions on the structure of the knowledge graph to achieve tractability. Due to this, most of the constraints and rules that we use for experimentation are not applicable to TML and its variants.

## 3.8. Conclusion

We have presented an approach for reasoning over uncertain temporal knowledge graphs based on probabilistic graphical models. We proposed a formal syntax and semantics for uncertain temporal knowledge graphs and formalized the MAP and conditional probability inference problems. We used Datalog-inspired constraints to detect erroneous facts in uncertain temporal knowledge graphs. Then, we applied MAP inference to obtain a most probable and conflict-free temporal knowledge graph from an uncertain, noisy one.

We evaluated the performance of the approach with two different solvers and over different datasets and showed that inference scales favorably, and that it handles noisy data gracefully, especially in the case of nRockIt. nRockIt, using MLNs, allows to use more expressive constraints than PSL (bodies not limited to conjunctive formulas; negative weights (cf. Section 3.2.3)). However, PSL scales better since it computes a soft approximation of the discrete MAP state, and it also allows to have a trade-off between precision and recall through the configurable activation threshold for atoms in the graph.

When focusing on precision and recall, we found that precision is impacted similar for nRockIt and PSL in the face of noisy data. However, nRockIt's recall is much less impacted by erroneous facts. This shows that it is well suited for our initial goal: increasing recall as compared to strict logical reasoning.

# 4 Scalable Probabilistic Query Answering and Logical Reasoning

## 4.1. Introduction

The Semantic Web community has brought up many standards – published as recommendations by the World Wide Web Consortium (W3C) – for knowledge representation and for querying knowledge bases using those. Especially the Resource Description Framework (RDF) (Schreiber and Raimond, 2014) and the Web Ontology Language (OWL) (Parsia et al., 2012) are mature formalisms for modeling knowledge that have gained a lot of traction, often in the form of knowledge graphs or Linked (Open) Data (Bizer et al., 2009). The SPARQL recommendation (The W3C SPARQL Working Group, 2013) defines an expressive query language for retrieving information from knowledge bases using those formalism.

Going further, the Semantic Web community has show a lot of interest in ontology-based data access (OBDA) (Calvanese et al., 2015). Especially the increase in efficiency of reasoning and query answering achieved through recent results on light-weight description logics provide new possibilities for using ontologies in data access. One approach for ontology-based data access is query rewriting, i.e. reformulating a given query using the background ontology in such a way that the resulting – more complex – query can directly be executed on a relational database. This is possible for different light-weight ontology languages, in particular the *DL-Lite* family (Artale et al., 2009).

At the same time, many applications in particular on the Semantic Web have to deal with uncertainty in the data and, as already argued by Paulheim and Pan (2012), optimizing the Semantic Web for precision alone is not sufficient. One can trivially design a system that achieves very high precision, but subsequently it will suffer from low recall. For IBM, the impact of this on linked data and ontologies was so severe, that they decided not to use structured knowledge bases for their famous Jeopardy-winning Watson system, as it would only have been able to answer about two percent of the posed questions (Kalyanpur et al., 2012). To improve recall and achieve a better overall performance, they developed a system that allows for imprecise or uncertain knowledge instead.

Handling uncertain or incomplete information is getting more and more important, not only in the domain of expert systems or query-answering systems like Watson, but also in other real world applications like data integration and information extraction.

- *Data integration* is the mapping of two or more heterogeneous data sources, e.g. ontologies without a common upper ontology or database with a different schema, into one combined knowledge base. In many cases it is not possible to find perfect matches between all concepts and properties of the different sources. However, quite often there are concepts that almost correspond to each other or have similar meanings. Mapping those with some specific confidence value yields an increased benefit for the resulting integrated knowledge base.

- *Information extraction* is another field where uncertain information plays an important role. Information extraction tries to generate meaningful, structured knowledge from merely unstructured data in the form of natural text. Although natural language processing provides good tools to derive information from unstructured text, natural language has many ambiguities that require context or background knowledge to discern the correct meaning of a term or statement (cf. (Theobald et al., 2013)).

Approaches combining logics with uncertain reasoning have a long history. Lukasiewicz and Straccia (2008) give an overview of the evolution of the field. More recent approaches include DISPONTE/BUNDLE (Riguzzi et al., 2013, 2015) or Pronto (Klinov and Parsia, 2013), which combine description logics with probabilistic reasoning, and Log-linear Description Logics (Niepert et al., 2011b). On the other hand, the logic programming and statistical relational learning community has developed probabilistic versions of Datalog-style languages (e.g. ProbLog (De Raedt et al., 2007)) that can be used to partially model ontological background knowledge. While for many of these languages efficient subsets have been identified (e.g. (Klinov and Parsia, 2013; Riguzzi et al., 2015)) and optimized reasoning algorithms have been proposed, none of the existing approaches is designed to handle a large amount of data as we find on the Web.

Conversely, the core idea of the ODBA approach builds on the concept of translating SPARQL queries to standard SQL queries which allows to handle large amount of data. This requires also to expand queries against the background knowledge that might be given in terms of a light-weight ontology. For its probabilistic extension the distinction between safe and unsafe queries is highly crucial. Research in probabilistic databases has shown that there is a strict dichotomy of safe (data complexity in PTIME) and unsafe (in #P-hard) queries (Suciu et al., 2011). Jung et al. have shown that query rewriting for OBDA can directly be lifted to the probabilistic case (Jung and Lutz, 2012). Furthermore, they prove that the complexity results and the dichotomy of safe and unsafe queries also carries over to probabilistic query answering in an OBDA setting. We will very briefly present the basic approach and the distinction between safe and unsafe queries with the help of an example from an information extraction scenario.

**Example 4.1** (Query Answering)
*Datasets extracted from Web pages by the Never Ending Language Learning Project (NELL (Carlson et al., 2010)) ReVerb (Fader et al., 2011) contain millions statements with associated probabilities, amongst others the following assertions about Arnold Schwarzenegger and his wife Maria Shriver.*

$$\begin{aligned}
&1.00\ \ \mathit{PoliticianUS}(\mathit{Arnold\_Schwarzenegger})\\
&1.00\ \ \mathit{Actor}(\mathit{Arnold\_Schwarzenegger})\\
&0.50\ \ \mathit{hasOffice}(\mathit{Arnold\_Schwarzenegger}, \mathit{President})\\
&1.00\ \ \mathit{husbandOf}(\mathit{Arnold\_Schwarzenegger}, \mathit{M\_Shriver})\\
&0.75\ \ \mathit{agentControls}(\mathit{NBC}, \mathit{M\_Shriver})\\
&1.00\ \ \mathit{acquired}(\mathit{NBC}, \mathit{Telemundo})
\end{aligned}$$

*While some of the statements are considered to be definitely true, some of them, however, are only believed to be true with a certain probability. Accessing the stored information in a meaningful way requires to query the knowledge base. In this work, we consider queries that allow us for example to ask for the probability that a politician has been president and actor or for the probability that a politician has been married to someone who is under control of a company. In terms of SPARQL, we can formalize these two queries as follows.*

```
ASK {
  ?x a :actor ;
     a :politician ;
       :hasOffice :president .
}
```

```
ASK {
  ?x a :actor ;
     a :politician ;
       :hasOffice :president .
       :spouse ?y .
  ?y :agentControlledBy ?z .
  ?z a :company .
}
```

*If we ask these queries directly to the given data, the resulting answers might not meet our expectations, because Arnold Schwarzenegger is, for example, not explicitly marked as a politician (but as a US politician). In particular, we can derive the following DL-Lite$_R$ axioms from the metadata in the NELL knowledge base, stating that US politicians are politicians, agentControlledBy is the inverse of agentControls and that the domain of the acquired relation is company:*

$$\begin{aligned}
\mathit{PoliticianUS} &\sqsubseteq \mathit{Politician}\\
\mathit{agentControlledBy} &\sqsubseteq \mathit{agentControls}^{-}\\
\exists\, \mathit{acquired} &\sqsubseteq \mathit{Company}
\end{aligned}$$

*Using these axioms, we can compute the probability that Arnold Schwarzenegger is a correct answer to the original SPARQL query by expanding it according to the given axioms.* △

Within this chapter we recall the formalism that enables us to translate and expand a given SPARQL query, based on the given set of *DL-Lite*$_R$ axioms such that we get a set of resulting SQL queries which can be applied to a data set stored in a relational database. If such an approach is applied and extended for a probabilistic setting, there are two crucial questions:

- Are the queries that we typically ask in a real-world scenario safe?
- And if yes, can we actually offer an efficient method to answer such queries?

These are the two research questions that we will answer within our this chapter. Starting with the first question, we note that the first query of our example is safe, while the second is unsafe. In order to understand the distribution of queries in a real world setting, we analyze the queries collected in the Linked SPARQL Queries Dataset (LSQ) (Saleem et al., 2015). LSQ contains queries which where posed against the public SPARQL endpoints of DBPedia[1], Linked Geo Data (LGD)[2], Semantic Web Dog Food (SWDF)[3], and the British Museum (BM)[4]. Although the queries in the dataset are not posed against probabilistic data, we argue that the large number of queries gives a good picture of the general information needs users have.

To answer the second question, we have developed a prototypical implementation that is designed to answer safe probabilistic SPARQL queries over large probabilistic knowledge bases up to several hundred millions of facts. This prototype is an implementation of the rules for deciding query safeness that have been proposed by Suciu et al. (Suciu et al., 2011). We report about its results applied to a synthetic benchmark dataset for probabilistic OBDA on the basis of the LUBM benchmark that can be scaled to an arbitrary number of probabilistic statements. We report also on its performance on a real world knowledge base (NELL), that was also the basis for the example we presented so far. We will also compare our results against another system that can be applied to answer probabilistic queries.

## 4.2. Preliminaries

In this section we describe how TIP-OWL (Definition 2.9) can be implemented through query rewriting on top of tuple-independent probabilistic databases. Then we explain extensional probabilistic query processing – the key to efficient query processing in probabilistic database – and give the definition for query safeness. This lays the foundation for tractable ODBA on top of probabilistic data.

[1]`http://dbpedia.org/`
[2]`http://linkedgeodata.org/`
[3]`http://data.semanticweb.org/`
[4]`http://bm.rkbexplorer.com`

### 4.2.1. Implementing Reasoning on Top of Probabilistic Databases

In this section, we first briefly recall the idea of first-order rewritability of queries in *DL-Lite* and then show that the query rewriting approach proposed by Calvanese et al. (Calvanese et al., 2007) can be used on top of tuple independent probabilistic databases for answering queries in TIP-OWL without changing the semantics of answers.

#### Query Rewriting

Query processing in *DL-Lite$_R$* is based on the idea of first-order reducibility. This means that for every conjunctive query $q$ we can find a query $q'$ that produces the same answers as $q$ by just looking at the ABox. Calvanese et al. also define a rewriting algorithm that computes a $q'$ for every $q$ by applying transformations that depend on the TBox axioms.

Given a consistent TBox $\mathcal{T}$ the algorithm takes a conjunctive query $q_0$ and expands it into a union of conjunctive queries $U$ starting with $U = \{q_0\}$. The algorithm successively extends $U$ by applying the following rule:

$$U = U \cup \{q[l/r(l,I)]\}$$

where q is a query in $U$, $l$ is a literal in $q$, $I$ is an inclusion axiom from $\mathcal{T}$ and $r$ is a replacement function that is defined as follows:

| **l** | **I** | $\mathbf{r}(l,I)$ |
|---|---|---|
| $A(x)$ | $A' \sqsubseteq A$ | $A'(x)$ |
| $A(x)$ | $\exists P \sqsubseteq A$ | $P(x,\_)$ |
| $A(x)$ | $\exists P^- \sqsubseteq A$ | $P(\_,x)$ |
| $P(x,\_)$ | $A \sqsubseteq \exists P$ | $A(x)$ |
| $P(x,\_)$ | $\exists P' \sqsubseteq \exists P$ | $P'(x,\_)$ |
| $P(x,\_)$ | $\exists P'^- \sqsubseteq \exists P$ | $P'(\_,x)$ |

| **l** | **I** | $\mathbf{r}(l,I)$ |
|---|---|---|
| $P(\_,x)$ | $A \sqsubseteq \exists P^-$ | $A(x)$ |
| $P(\_,x)$ | $\exists P' \sqsubseteq P^-$ | $P'(x,\_)$ |
| $P(\_,x)$ | $\exists P'^- \sqsubseteq \exists P^-$ | $P'(\_,x)$ |
| $P(x,y)$ | $P' \sqsubseteq P$ | $P'(x,y)$ |
| $P(x,y)$ | $P'^- \sqsubseteq P^-$ | $P'(x,y)$ |
| $P(x,y)$ | $P' \sqsubseteq P^-$ | $P'(y,x)$ |
| $P(x,y)$ | $P'^- \sqsubseteq P$ | $P'(y,x)$ |

Here '_' denotes an unbound variable, i.e., a variable that does not occur in any other literal of any of the queries.

**Definition 4.1** (Derived Query)
*Let $Q = L_1 \wedge \cdots \wedge L_m$ be a conjunctive query over a DL-Lite terminology. We write $Q \xrightarrow{r} Q'$ if $Q' = Q \vee Q[L_i/r(L_i,I)]$ for some literal $L_i$ from $Q$. Let $\xrightarrow{r}^*$ denote the transitive closure of $\xrightarrow{r}$, then we call every $Q'$ with $Q \xrightarrow{r}^* Q'$ a derived query of $Q$. $Q'$ is called maximal if there is no $Q''$ such that $Q \xrightarrow{r}^* Q''$ and $Q' \xrightarrow{r}^* Q''$.* $\bigcirc$

Using the notion of a maximal derived query, we can establish the FOL-reducibility of *DL-Lite* by rephrasing the corresponding theorem by Calvanese et al. (Calvanese et al., 2007).

**Theorem 4.1** (FOL-Reducibility of *DL-Lite* (Calvanese et al., 2007))
*Query answering in DL-Lite*$_R$ *is FOL-reducible. In particular, for every query* $Q$ *with maximal derived query* $Q'$ *and every DL-Lite*$_R$ *TBox* $\mathcal{T}$ *and non-empty ABox* $\mathcal{A}$ *we have* $\mathcal{T} \cup \mathcal{A} \models Q$ *if and only if* $\mathcal{A} \models Q'$.

**Example** We illustrate the rewriting using the queries and the three ontological axioms from the motivating example. For the first query, the subclass relation between *politicianus* and *politician* triggers a rewriting leading to the new query $Q'(X) \Leftarrow \mathit{politicianus}(X), \mathit{actor}(X), \mathit{hasoffice}(X, \mathit{president})$ which has to be united with the original query to form the new query $Q \vee Q'$. This query can be simplified to:

$$\begin{gathered}(\mathit{politician}(X) \vee \mathit{politicianus}(X)) \\ \wedge\ \mathit{actor}(X) \wedge \mathit{hasoffice}(X, \mathit{president})\end{gathered}$$

For the second query, the rewriting is slightly more complicated, but still can be computed in a single go through the query. Each of the axioms triggers a rewriting. Besides the rewriting already performed for the first query, the literal $\mathit{agentcontrolledby}(Y, Z)$ can be rewritten to $\mathit{agentcontrols}(Z, Y)$. Secondly, the literal $\mathit{company}(Z)$ can be rewritten to $\mathit{acquired}(Z, \_)$, after simplification leading to the following rewritten query:

$$\begin{gathered}(\mathit{politician}(X) \vee \mathit{politicianus}(X)) \\ \wedge\, \mathit{spouse}(X, Y) \\ \wedge\, (\mathit{agentcontrolledby}(Y, Z) \vee \mathit{agentcontrols}(Z, Y)) \\ \wedge\, (\mathit{company}(Z) \vee \mathit{acquired}(Z, \_))\end{gathered}$$

The resulting query can now directly be executed on the probabilistic database.

### Correctness of Query Processing

Implementing TIP-OWL on top of probabilistic databases can now be done in the following way. The ABox is stored in the probabilistic database. It is easy to see that the probabilistic semantics of TIP-OWL ABoxes and tuple independent databases coincide. What remains to be done is to show that the idea of FOL-reducibility carries over to our probabilistic model. In particular, we have to show that a rewritten query has the same probability given a knowledge base with empty TBox as the original query given a complete knowledge base. This result is established in the following corollary.

**Corollary 4.1**
*Let $\mathcal{TKB} = \langle \mathcal{T}, \mathcal{A}, \mathcal{P} \rangle$ be a TIP-OWL knowledge base and $\mathcal{TKB}' = \langle \emptyset, \mathcal{A}, \mathcal{P} \rangle$ the same knowledge base, but with an empty TBox. Let further $Q$ be a conjunctive query and $Q'$ a union of conjunctive queries obtained by rewriting $Q$ on the basis of $\mathcal{T}$, then*

$$P(Q|\mathcal{TKB}) = P(Q'|\mathcal{TKB}')$$

*Proof.* The result directly follows from Theorem 4.1 and the definition of the probability of queries given above. From Theorem 4.1 we get that $P(Q|\mathcal{KB})$ does not change as $\mathcal{T}, \mathcal{A} \models Q$ if and only if $\mathcal{A} \models Q'$. Further, as $P(\mathcal{KB}|\mathcal{TKB})$ only depends on $\mathcal{A}$ this part also stays unchanged. □

This result provides us now with a framework for query answering over probabilistic data using a logical background theory. A question that remains to be answered is about the efficiency of the approach. We know that query rewriting is in PTIME (Calvanese et al., 2007); and that probabilistic query answering for safe queries is in PTIME (Dalvi and Suciu, 2012). These results suggest that query answering in our model is also in PTIME as long as we stick to certain types of queries. In particular, we know that query answering is in PTIME if the rewritten query is safe. To ensure efficiency, we characterize queries whose rewritten versions are safe in the next section.

### 4.2.2. Complexity of Query Processing in TIP-OWL

Over the past decade, the database community has developed efficient methods for querying uncertain information in probabilistic databases. Important results are the introduction of the independent tuple model for probabilistic data as well as a complete characterization of queries that can be computed in polynomial time. In this section, we briefly review these results as a basis for our extension towards ontological background knowledge.

#### Extensional Query Processing

Answering arbitrary (unions of) conjunctive queries over probabilistic databases typically requires the construction of the complex event description that represents possible answers as a basis for computing their probability (Suciu et al., 2011). This approach, that is often called intensional query processing, is inherently intractable as it corresponds to enumerating all possible worlds. It has been shown that the key to efficient query processing in probabilistic databases is to avoid the computation of complex event descriptions and to directly compute the probability of a complex query from the probabilities of subqueries. This approach, that is referred to as extensional query processing, has been shown to correctly compute the probability of queries for the class of tractable queries.

Extensional query processing for unions of conjunctive queries proceeds in a number of steps that each reduces the problem of computing the probability of a query to computing and aggregating the probability of smaller queries. If the steps succeed, the query can be computed in polynomial time of the size of the database. Those steps are given by Suciu et al. in the form of six rules (Suciu et al., 2011). To decide whether a rule is applicable to a query or not we need two more definitions that specify certain characteristics of a query.

**Definition 4.2** (Syntactic Independence (Suciu et al., 2011))
*Two queries $Q_1$ and $Q_2$ are* syntactically independent *if no two atoms from each query unify. Atoms are unifiable if they become the same tuple when instantiated.* ○

To be unifiable, two atoms must use the same relation symbol. An example are the atoms $hasOffice(x, President)$ and $hasOffice(Trump, President)$ as they become the same tuple when $x$ is instantiated with *Trump*. Whereas the two atoms $hasOffice(Obama, y)$ and $hasOffice(Trump, y)$ are not unifiable as they can never become the same tuple. Furthermore, if two queries are syntactically independent, they are also independent probabilistic events.

**Definition 4.3** (Separator Variable (Suciu et al., 2011))
*Let $Q$ be a query of the form $Q = \exists x.Q'$. The variable $x$ is called a* separator variable *if for any two unifiable atoms in $Q$, $x$ occurs in the same position.* ○

If we have a query with separator variable $x$, its instantiations with two different constants are syntactically independent. To obtain separator variables often the following equivalence is applied: $\exists x_1 Q_1 \vee \exists x_2 Q_2 \equiv \exists x.(Q_1[x/x_1] \vee Q_2[x/x_2])$.

Using those two concepts, we can now explain the six rules to determine query safeness.

**Definition 4.4** (Six Rules for Query Safeness (Suciu et al., 2011))

**Rule 1 (*Independent Join*):** If the query can be written as $Q = Q_1 \wedge Q_2$ with $Q_1$ and $Q_2$ being two syntactically independent subqueries then

$$P(Q_1 \wedge Q_2) = P(Q_1) \cdot P(Q_2)$$

**Rule 2 (*Independent Project*):** Making sure that instantiations are independent requires to find a so-called *separator variable*, a variable that occurs at the same position in every conjunct.

Finding separator variables might require to re-rank predicates to move variables in the right position and to replace variables in different disjuncts using the equivalence $\exists x_1 Q_1 \vee \exists x_2 Q_2 \equiv \exists x.(Q_1[x/x_1] \vee Q_2[x/x_2])$.

If a query $Q$ can be written as $\exists x.Q$ where x is a separator variable then

$$P(\exists x.Q) = 1 - \prod_{a \in ADom} (1 - P(Q[a/x]))$$

The instantiated subqueries are either single literals – in this case their probability can be read directly from the database – or they are queries in DNF. In this case processing continues until the query has been completely broken down. This rule is the critical one that determines whether extensional query processing is possible or not. It requires instantiating the query with different values from the domain *ADom* of the corresponding attribute. If we cannot make sure that different instantiations are independent via a separator variable, processing stops here and probabilities cannot be computed efficiently.

**Rule 3 (*Independent Union*):** If the query $Q$ can be written as $Q = Q_1 \vee Q_2$ with $Q_1$ and $Q_2$ being syntactically independent then

$$P(Q_1 \vee Q_2) = 1 - (1 - P(Q_1)) \cdot (1 - P(Q_2))$$

**Rule 4 (*Negation*):** If the query is if the form $\neg Q$ then

$$P(\neg Q) = 1 - P(Q)$$

**Rule 5 (*Inclusion-Exclusion Formula*)** In the general case, the queries $Q'_i$ are of the form $Q_{i_1} \wedge \cdots \wedge Q_{i_k}$ and the different subqueries are not independent. The probability of these queries can be computed using the dual of the well-known inclusion-exclusion rule:

$$P(Q_1 \wedge \ldots \wedge Q_n) = - \sum_{s \subseteq [n], s \neq \emptyset} (-1)^{|s|} P\left(\bigvee_{i \in s} Q_i\right)$$

While the direct application of this rule may generate intractable subqueries whose probability is part of the formula, Dalvi and Suciu have shown that these intractable subqueries do not have to be computed, because they cancel each other out in the final summation. They also show that the Moebius function over lattices can be used to only generate the relevant summands in the first place, thus avoiding unnecessary computation or detection of hard queries (Dalvi and Suciu, 2012).

**Rule 6 (*Attribute Ranking*):** Ranking is used to partition the data according to a predicate, to facilitate the search for separator variables. There are two types of ranking: attribute-attribute-ranking where two attributes are compared, and attribute-constant-ranking comparing an attribute with a constant (attributes are simply characterized by position in the respective predicate).

Attribute-attribute-ranking is done as follows: For some predicate $R$ with attributes $A$ and $B$, we partition the data by the use of three new predicates $R_1 = \sigma_{A<B}(R)$,

$R_2 = \sigma_{A=B}(R)$ and $R_3 = \sigma_{A>B}(R)$, and replace all occurrences of $R$ with $R_1 \vee R_2 \vee R_3$. The resulting query is denoted $Q^r$.

Attribute-constant-ranking is done as follows: For some predicate $R$ with attribute $A$, if there are two unifiable atoms where one has a variable at position $A$ and the other some constant $c$, the data is partitioned by the new predicates $R_1 = \sigma_{A \neq c}(R)$ and $R_2 = \sigma_{A=c}(R)$, and again substituting $R$ with $R_1 \vee R_2$ and denoting the new query as $Q^r$.

Thus, for any query $Q$,

$$P(Q) = P(Q^r)$$

with $Q^r$ being the ranked query.

Note that by ranking predicates in this way, we create three syntactically independent predicates (they use different predicate symbols and also do not share assignments), which in turn ideally makes more subqueries syntactically independent.

$\bigcirc$

Queries that can be fully processed using the rules above – i.e. they are completely broken down into single literals that can be looked up in the database – are called *safe*. By applying those rules in a systematic way, we get an algorithm which can compute the probability of any safe query. This algorithms main function, which recursively computes $P(Q)$, is shown in Algorithm 1.

**Algorithm 1** Extensionally compute P(Q) (from (Dalvi and Suciu, 2012))

```
Require: A ranked, conjunctive query Q in CNF; a tuple independent probabilistic database
 1: Q is a conjunctive query in CNF: Compute symbol components $Q = Q_1 \wedge \cdots \wedge Q_m$
 2: if $m \geq 2$ then
 3:     return $\prod_{i=1,\cdots,m} P(Q_i)$
 4: end if
 5:
 6: $Q = Q_1 \wedge \cdots \wedge Q_m$ is a symbol-connected query in CNF:
 7: if $m \geq 2$ then
 8:     return $-\sum_{s\subseteq[n],s\neq\emptyset}(-1)^{|s|} P\left(\bigvee_{i\in s} Q_i\right)$
 9: end if
10:
11: Q is a disjunctive query: Compute symbol components $Q = Q_1 \vee \cdots \vee Q_m$
12: if $m \geq 2$ then
13:     return $1 - \left(\prod_{i=1,\cdots,n} 1 - P(Q_i)\right)$
14: end if
15:
16: if $Q$ has a separator variable z then
17:     return $1 - \left(\prod_{a\in ADom} 1 - P(Q[a/x])\right)$
18: else
19:     return false
20: end if
```

The algorithm starts with a fully attribute-constant ranked query over some probabilistic database. Then it tries to apply the rules in the following order (whenever required, the query is rewritten from DNF to CNF, or vice versa, by applying the distributivity law):

1. Rule 1: Independent join (Line 1)
2. Rule 5: Inclusion-exclusion formula (Line 6)
3. Rule 3: Independent union (Line 11)
4. Rule 2: Independent project (Line 15)
5. Repeat from Step 1, if the query still contains variables and the algorithm recursed in this iteration

When searching for a separator variable, we also repeatedly apply attribute-attribute ranking (Rule 6) to some pairs of attributes, if no separator is found in the initial try. Whenever we reach an atom that does not contain variables (i.e. only constants or projected-out separator variables), we look up its probability in the database. Similar, we apply Rule 4 (Negation) as suitable throughout the algorithm.

Each recursion of the algorithm processes a simpler subexpression, until the probability of an individual can be read directly. If the algorithm ever gets stuck and cannot simplify the query further, it is technically because of not finding a separator variable.

We explain the application of the rules on the example of two queries; one safe and the other unsafe.

**Example 4.2** (Extensional Processing of a Safe Query)
*Consider the following conjunctive boolean query:*

$$Q \Leftarrow \exists x.\exists y.\exists z.(R(x,y) \land R(z,y))$$

*A concrete version of this query can example ask whether we know of a person that was head of state in two different countries:* $Q \Leftarrow \exists x.\exists y.\exists z.(\mathit{hasHeadOfState}(x,y) \land \mathit{hasHeadOfState}(z,y))$.

*Deciding whether the query is safe and calculating its probability using the six rules of Suciu et al. (2011) is done as follows:*

$$\begin{aligned}
P(Q) &= P\big(\exists x.\exists y.\exists z.(R(x,y) \land R(z,y))\big) && \text{[equivalence]}\\
&= P\Big(\exists y.\big(\exists x.R(x,y) \land \exists z.R(z,y)\big)\Big) && \text{[independent project]}\\
&= 1 - \prod_{a \in ADom} \Big(P\big(\exists x.R(x,[a/y]) \land \exists z.R(z,[a/y])\big)\Big)
\end{aligned}$$

*Let* $ADom = \{a_1, \ldots, a_n\}$ *and* $Q'(a_i) = \exists x.R(x,a_i) \land \exists z.R(z,a_i)$.

$$\begin{aligned}
P(Q'(a_i)) &= P\big(\exists x.R(x,a_i) \land \exists z.R(z,a_i)\big) && \text{[inclusion-exclusion formula]}\\
&= P\big(\exists x.R(x,a_i)\big)\\
&\quad + P\big(\exists z.R(z,a_i)\big)\\
&\quad - P\big(\exists x.R(x,a_i) \lor \exists z.R(z,a_i)\big)
\end{aligned}$$

*Let* $Q''_1 = \exists x.R(x,a_i)$, $Q''_2 = \exists z.R(z,a_i)$, $Q''_3 = \exists x.R(x,a_i) \lor \exists z.R(z,a_i)$.

$$\begin{aligned}
P(Q''_1(a_i)) &= P\big(\exists x.R(x,a_i)\big) && \text{[independent project]}\\
&= 1 - \prod_{b \in ADom} \big(1 - P(R([b/x],a_i))\big)\\
P(Q''_2(a_i)) &= P\big(\exists z.R(z,a_i)\big) && \text{[independent project]}\\
&= 1 - \prod_{c \in ADom} \big(1 - P(R([c/x],a_i))\big)\\
P(Q''_3(a_i)) &= P\big(\exists x.R(x,a_i) \lor \exists z.R(z,a_i)\big) && \text{[equivalence]}\\
&= P\Big(\exists x'.\big(R([x'/x],a_i) \lor R([x'/z],a_i)\big)\Big) && \text{[independent project]}\\
&= 1 - \prod_{d \in ADom} \Big(1 - P\big(R([d/x'],a_i) \lor R([d/x'],a_i)\big)\Big)
\end{aligned}$$

*Now, every variable in the original query is projected out, and the probability of every atom can be read from the database. Putting all parts of the query together, the final formula to compute the probability is as follows:*

$$\begin{aligned}
P(Q) &= 1 - \prod_{a \in ADom} \Big( \quad P(Q'(a)) \Big) \\
&= 1 - \prod_{a \in ADom} \Big( \quad P(Q''_1(a)) + P(Q''_2(a)) - P(Q''_3(a)) \Big) \\
&= 1 - \prod_{a \in ADom} \Big( \quad \Big(1 - \prod_{b \in ADom} (1 - P(R(b,a)))\Big) \\
&\qquad\qquad + \Big(1 - \prod_{c \in ADom} (1 - P(R(c,a)))\Big) \\
&\qquad\qquad - \Big(1 - \prod_{d \in ADom} \Big(1 - P\big(R(d,a) \vee R(d,a)\big)\Big)\Big)\Big) \\
&= 1 - \prod_{a \in ADom} \Big(1 - \prod_{b \in ADom} (1 - P(R(b,a))) \\
&\qquad\qquad - \prod_{c \in ADom} (1 - P(R(c,a))) \\
&\qquad\qquad + \prod_{d \in ADom} \Big(1 - P\big(R(d,a) \vee R(d,a)\big)\Big)\Big) \\
&= 1 - \prod_{a \in ADom} \Big(1 - \prod_{d \in ADom} \Big(1 - P\big(R(d,a)\big)\Big)\Big)
\end{aligned}$$

△

It has been shown that safe queries exactly correspond to queries that can be computed in PTime whereas queries that cannot be completely processed using these steps – in particular queries for which we cannot find a separator variable – are #P-hard. This means that we can use the notion of safeness and the processing rules above to analyze the general complexity of certain classes of queries.

**Definition 4.5** (Safeness)
*A query Q is called* safe *if and only if it can be computed by iteratively applying the six rules in Definition 4.4, resulting in a fully instantiated query.* ○

**Theorem 4.2** (Dichotomy Theorem (adapted from (Suciu et al., 2011)))
*The probability of safe queries can be computed in PTIME. If a query is not safe computing its probability is #P-hard.*

Jung et al. proved that Theorem 4.2 also holds for probabilistic OBDA in OWL 2 QL (Jung and Lutz, 2012). This essentially means that the same dichotomy of safe and unsafe queries still applies under the presence of reasoning through query rewriting.

Table 4.1.: Overview of queries in the LSQ dataset

| | SELECT | ASK | DESCRIBE | CONSTRUCT | Total |
|---|---|---|---|---|---|
| SWDF | 58 741 | 157 | 26 533 | 52 | 85 483 |
| DBpedia | 736 726 | 37 174 | 777 | 7 613 | 782 290 |
| LGD | 265 410 | 25 140 | 24 | 7 004 | 297 578 |
| BM | 29 073 | 0 | 0 | 0 | 29 073 |
| Total | 1 089 950 | 62 471 | 27 334 | 14 669 | 1 194 424 |

## 4.3. Analysis of the SPARQL Dataset and Query Safeness

After establishing the basis for logical and probabilistic query answering, we now analyze a large corpus of real-world SPARQL queries for their complexity in query answering. The goal is to determine the amount of safe and unsafe queries in real-world scenarios, and thus show the general feasability of probabilistic logical query answering over large data in practice. SPARQL specified as W3C recommendation and plays a comparable role for the Semantic Web as SQL in relational databases. For our analysis we used the queries collected in the Linked SPARQL Queries Dataset (LSQ) (Saleem et al., 2015). LSQ contains queries which where posed against the public SPARQL endpoints of DBPedia[5], Linked Geo Data (LGD)[6], Semantic Web Dog Food (SWDF)[7], and the British Museum (BM)[8]. Although the queries in the dataset are not posed against probabilistic data, we argue that the large number of queries gives a good picture of the general information needs users have. In the case of DBpedia, there is actually some amount of uncertainty in its generation, as the data is automatically extracted from Wikipedia, which itself can contain uncertain/wrong information.

Table 4.1 shows the different query types and their distribution in the different datasets. Those numbers contain only queries that parse successfully; LSQ also lists queries with parse errors. They clearly show that the main amount of queries are SELECT queries. Interestingly, the dataset for the British Museum's endpoint only consists of SELECT queries; all being uniformly structured. This suggests that those are not actual user queries but automatically created by some system. However, the queries from the British Museum make up only a small part of the whole LSQ dataset.

Looking at the used SPARQL features, FILTER and DISTINCT are the most prominent, with UNION coming in second. GROUP BY and EXISTS play only a minor role. However, note that DISTINCT is technically a GROUP BY over all variables in the SELECT clause.

[5]`http://dbpedia.org/`
[6]`http://linkedgeodata.org/`
[7]`http://data.semanticweb.org/`
[8]`http://bm.rkbexplorer.com`

Table 4.2.: Used SPARQL features

| | UNION | FILTER | DISTINCT | GROUP BY | EXISTS |
|---|---|---|---|---|---|
| BM | 0 | 0 | 29073 | 0 | 0 |
| DBpedia | 36127 | 183882 | 144245 | 3 | 3 |
| LGD | 28827 | 92558 | 66206 | 11 | 0 |
| SWDF | 27982 | 818 | 39000 | 445 | 123 |
| Total | 92936 | 277258 | 278524 | 459 | 126 |

The focus of our analysis is on SELECT and ASK queries. For DESCRIBE queries there is no formal definition of how a result is produced, thus making it impossible to determine if the process is safe; whereas CONSTRUCT queries do not directly translate to query answering as they generate a new graph and not a result set. Furthermore, our current implementation cannot analyze queries containing features, like GROUP BY, DISTINCT, EXISTS, OPTIONAL, and endpoint specific functions not part of the SPARQL recommendation (e.g. Oracle's version of COUNT). After removing those we are left with 507 241 queries for which we can determine query safeness.

We used parts of Ontop (Calvanese et al., 2017), a system for deterministic OBDA, to translate the SPARQL queries to (union of) conjunctive queries. Those queries were then processed by Algorithm 1 to determine their safeness. The implementation is written in Java and the experiments were run on an Intel Core i5@2.9 GHz with 4GB of RAM for the Java VM.

The distribution of safe and unsafe queries is shown in Table 4.3. With 99.68%, almost all of the *analyzed* queries are safe. Regarding the total number of 1 194 424 queries, 42.33% are definitly safe. Note that this does not imply that 57.67% of the queries are unsafe. Those queries need a more thorough analysis because of their usage of DISTINCT/GROUP BY, FILTER, HAVING, EXISTS, or OPTIONAL.

Looking at the numbers for the different data sources, the percent of unsafe queries sent to the Semantic Web Dog Food endpoint is significantly higher than for the other two. Analyzing those queries we found the major number being of the form

```
SELECT ?targetType
WHERE { ?obj a <someURI>.
        ?obj <someOtherURI> ?targetObj.
        ?targetObj a ?targetType.
}
```

This suggest that those queries are generated by some tool and not by a user. Overall, this shows that the users' general information needs can also be satisfied over probabilistic data in a tractable way.

Table 4.3.: Analysis results for queries in the LSQ dataset

| | DBpedia | | LGD | | SWDF | | Total | |
|---|---|---|---|---|---|---|---|---|
| | # | % | # | % | # | % | # | % |
| Safe | 287 487 | 99,72 | 199 636 | 99,79 | 18 240 | 97,93 | 505 563 | 99,68 |
| Unsafe | 809 | 0,28 | 414 | 0,21 | 386 | 2,07 | 1 609 | 0,32 |
| Total | 288 296 | | 200 050 | | 18 626 | | 507 172 | |

Our implementation of the algorithm is still quite simple and we are not able to logically simplify all queries. Thus determining their safeness becomes sometimes unfeasible. This results in a timeout for 69 queries after 10 minutes. Still, the average processing time for a query is 86ms.

## 4.4. Benchmarks on Probabilistic Data

In Section 4.3 we have shown that a significant amount of real-world queries are safe and can in theory be answered efficiently. This section complements those results by benchmarking the performance of probabilistic logical reasoning on two different datasets. We first describe the two used datasets and then show the results of the experimental evaluation in the following section.

### 4.4.1. Benchmark Data for Probabilistic OBDA

We use two different datasets for our experimental evaluation. The first dataset is the ontology and knowledge base created by NELL, which presents a large real-world dataset consisting of uncertain data. Second, to assess the scalability of the approach, we created a modified version of the Lehigh University Benchmark (LUBM), a synthetic benchmark for OWL reasoners that generates datasets of various sizes. We did not use one of the datasets from which LSQ collected the queries, as it would have been hard to scale them to different sizes and still ensure meaningful query results.

#### NELL

NELL is an Open Information extraction system that extracts facts from text found in a large corpus of web pages. As a result, NELL generates triples like `wifeof(katie_ holmes, tom_cruise)`, called candidate beliefs, that are annotated with different levels of confidence in terms of a number in the range $(0, 1]$.

Within the context of our approach these candidate beliefs form the ABox of our *TIP-OWL* knowledge base, while the confidences are interpreted as probabilities. NELL organizes extracted facts in a terminology consisting of concepts (called categories in

| Dataset | Assertions |
|---|---|
| NELL full | 2 259 750 |
| NELL filtered | 467 943 |

| Dataset | Assertions | % distinct |
|---|---|---|
| LUBM 1 | 717 250 | 54.77 |
| LUBM 10 | 7 232 663 | 55.69 |
| LUBM 100 | 71 698 666 | 55.66 |
| LUBM 200 | 143 311 100 | 55.67 |
| LUBM 500 | 361 432 844 | 55.12 |
| LUBM 1000 | 719 097 512 | 55.32 |

Table 4.4.: Size of the different NELL and LUBM datasets.

NELL context) and roles (relations) and specifies domain and range restrictions, property symmetry, and disjointness of concepts and properties. The *DL-Lite*$_R$ fragment of this terminology is used as TBox of our *TIP-OWL* knowledge base. We use the high confidence knowledge base of NELL (iteration 860) which contains only facts with a score of at least 0.75. It contains 2.3 million extracted facts about 1.8 million objects as compared to the full dataset with roughly 50 million. The TBox, which is the same for all datasets, consists of 558 concepts, 1 255 properties, and 5 132 axioms (domain and range restrictions, property symmetry, concept and property disjointness).

To show the benefits and the scalability of our approach, we defined the following queries that are posed against the *TIP-OWL* version of NELL.

$$
\begin{aligned}
Q_A(X) &\Leftarrow person(X)\\
Q_B(X) &\Leftarrow person(X), bornin(X, paris)\\
Q_C(X) &\Leftarrow book(X), movie(X)\\
Q_D(Z) &\Leftarrow hasParent(X, Y), hasparent(Y, Z)\\
Q_E(X) &\Leftarrow actor(X), directordirectedmovie(X, Y), writerwrotebook(X, Z)\\
Q_F(X) &\Leftarrow politician(X), actor(X), hasoffice(X, president)
\end{aligned}
$$

We use this dataset mainly to investigate the benefits of using background knowledge and reasoning on top of probabilistic data in terms of increased recall.

### Probabilistic LUBM

The Lehigh University Benchmark (LUBM) (Guo et al., 2005) is a well known and widely used benchmark for OWL-based reasoning systems. Lutz et al. published a *DL-Lite*$_R$ version of LUBM (Lutz et al., 2013). Additionally to restricting the expressivity of the ontology, they modified it to make it more suitable in an OBDA setting: First, they added multiple concept inclusions with existential restriction on the right hand side, and second they extended the class hierarchy to be closer to real-world ontologies in its size.

We chose to extend LUBM over other synthetic benchmarks like SP$^2$Bench (Schmidt et al., 2009), BSBM (Bizer and Schultz, 2001), or FishMark (Bail et al., 2012). SP$^2$Bench and BSBM only provide a very simple or no ontology, but rather focus on complex queries. FishMark contains an expressive ontology more suitable for our evaluation. However, it does not provide a generator for datasets of various sizes. Lutz et al.'s version of the LUBM ontology is of sufficient complexity to evaluate the scalability of reasoning in an OBDA setting, and it offers the possibility to generate datasets of different sizes.

The benchmark dataset for probabilistic OBDA is generated in two steps:

1. We extended the generator to create probabilistic ABoxes.
2. To increase the complexity of the probabilistic reasoning, we created redundancies in the dataset.

In the first step we extended the implementation of the data generator to attach probabilities to every ABox axiom. Those probabilities are randomly distributed in (0,1]. We did not include a fixed percentage of certain axioms. The generator thus creates datasets of various size with probabilistic axioms. However, each axiom is contained exactly once, thereby trivializing the calculation of the final probabilities in a result set. To alleviate this, we used the option to change the seed that LUBM uses to determine the number of instances of departments, professors, students, etc.

For every dataset size, we generated five ABoxes each with a different seed (0, 1, 42, 776, 141984). Combined, these five ABoxes serve as one probabilistic benchmark dataset. Note that our probabilistic version of LUBM is roughly five times larger than the normal LUBM of the same size, i.e. LUBM 1 contains only one university, whereas probabilistic LUBM 1 contains five different versions of that university, with different numbers of departments, professors, students, etc.

In the evaluation we used those queries of the original LUBM benchmark for which probabilities can be computed efficiently, i.e. queries 1, 3–6, and 10–14:

$$\begin{aligned}
Q_1(X) &\Leftarrow takesCourse(X, univ0\_dept0), type(X, graduateStudent)\\
Q_3(X, Y_1, Y_2, Y_3) &\Leftarrow publicationAuthor(X, univ0\_asstProf0), type(X, publication)\\
Q_4(X) &\Leftarrow worksFor(X, univ0\_dept0), name(X, Y_1), emailAddress(X, Y_2),\\
&\qquad telephone(X, Y_3), type(X, professor)\\
Q_5(X) &\Leftarrow memberOf(X, univ0\_dept0), type(X, person)\\
Q_6(Z) &\Leftarrow type(X, student)\\
Q_{10}(X) &\Leftarrow takesCourse(X, univ0\_graduateCourse0), type(X, student)\\
Q_{11}(X) &\Leftarrow subOrgOf(X, univ0), type(X, researchGroup)\\
Q_{12}(X, Y) &\Leftarrow worksFor(X, Y), type(X, chair), subOrgOf(Y, univ0),\\
&\qquad type(Y, department)\\
Q_{13}(X) &\Leftarrow hasAlumnus(univ0, X), type(X, person)\\
Q_{14}(X) &\Leftarrow type(X, undergraduateStudent)
\end{aligned}$$

$Q_{11}$ and $Q_{12}$ require the reasoner to handle the transitive object property `subOrgOf`. However, transitive object properties are not allowed in *DL-Lite*$_R$. To circumvent this and still be able to use this query in the evaluation, we manually extended those queries to handle transitivity up to the maximum depth occurring in the data (in this case 2).

### 4.4.2. Experimental Evaluation

We evaluate our implementation of query processing for *TIP-OWL* on the two datasets presented in the previous section. Our main goal is to show that our implementation scales to very large ABoxes and outperforms existing methods on safe queries.

#### Setting

Within our experiments we focus on answering the following two questions:

1. What are the benefits of exploiting the TBox by using it in the query rewriting process for a dataset like NELL?
2. How well does our algorithm scale with respect to different types of queries and subsets of NELL and a probabilistic LUBM, and compared to another system?

For answering the first question, we compare query results with and without query rewriting. We expect that rewriting the queries yields larger result sets. In particular, we expect that many interesting results are missed out when we ask the query directly without any expansion.

For answering the second question, we compare our implementation against the ProbLog system (De Raedt et al., 2007), which also uses the independent tuple semantics. While ProbLog does not support the complete expressivity of *DL-Lite*[9], it is sufficient to formulate and answer all safe conjunctive query of the dataset. For the comparison with ProbLog, we used a subset of the data consisting of about half a million facts. As ProbLog does not support SPARQL, the queries where modeled as concepts of the knowledge base, and results for that concept were computed with ProbLog. For example, the SPARQL query

```
SELECT ?x {
?x a :actor ;
a :politician;
:hasOffice :president . }
```

is modeled as the concept *politicianActor* for which instances and probabilities can be computed:

$$\mathit{politicianActor}(X) \Leftarrow \mathit{politician}(X), \mathit{actor}(X), \mathit{hasOffice}(X, \mathit{president})$$

[9]In particular, axioms of the form $A \sqsubseteq \exists R$ cannot be represented in ProbLog.

| | Plain | | Rewritten | |
|---|---|---|---|---|
| | # res. | # pred. | # res. | # pred. |
| $Q_A$ | 5 405 | 1 | 319 986 | 148 |
| $Q_B$ | 1 | 2 | 4 | 152 |
| $Q_C$ | 352 | 2 | 414 | 12 |
| $Q_D$ | 0 | 2 | 80 | 40 |
| $Q_E$ | 0 | 3 | 1 | 11 |
| $Q_F$ | 2 | 4 | 14 | 29 |

Table 4.5.: Number of results with and without reasoning, and increase in query size (predicates)

Additionally, to assess the general scalability of the approach, we run our implementation on different sizes of probabilistic LUBM, i.e. 1, 10, 100, 200, 500, and 1000 universities.

Experiments were run on a virtual machine with 4 cores (2.4GHz) and 16GB RAM running Ubuntu 14.10 Server. We used PostgreSQL 9.4 64bit and ProbLog 2. We used the default settings of the database and did no special tuning apart from increasing the available RAM. The NELL dataset was loaded into a single table. The LUBM datasets use different tables for class, object, and data property assertions, one of each for different sizes of the benchmark. Query rewriting was done manually at this point, but we do not expect a significant impact on this step on the overall performance. As ProbLog always has to load all the data and does not provide persistent storage like a database system, we measured the time ProbLog takes to parse the file without a query and subtracted that amount from the query time.

An existing probabilistic database like MayBMS was not used as those were not able to parse and process the complex structure of JOINs, UNIONs, and sub-queries produced by the query rewriting, resulting in various error messages.

### NELL Dataset

Table 4.5 shows the results of comparing query answering with and without rewriting. The number of results generally increases – sometimes dramatically (cf. $Q_A$) – and we can even find answers to $Q_D$ which produced no results without rewriting. The large increase in results for $Q_A$ is due to *person* being a very general concept of the NELL hierarchy and most instances are described using more specific concepts. $Q_D$ has no answers without rewriting because the relation *personhasparent* is never used, but only its inverse *parentofperson*.

Exploiting the TBox often changes the probabilities for an individual answer to a query as new evidence is added to the computation. For example the probability for *concept:person:sandy* being a *person* in $Q_A$ or $Q_B$ increase from 0.96875 to 1.0. The knowledge base only states that Sandy is a *person* with probability 0.96875. Through the

| | full | filtered |
|---|---|---|
| ProbLog | 24.393 sec | 5.383 sec |
| SQL Loading | 193.040 sec | 12.163 sec |
| SQL Indexing | 1 007.291 sec | 47.427 sec |

Table 4.6.: Dataset loading times (sec)

| | $Q_A$ | $Q_B$ | $Q_C$ | $Q_D$ | $Q_E$ | $Q_F$ |
|---|---|---|---|---|---|---|
| ProbLog (filtered) | - | 97.667 | 1.812 | - | 2.673 | 9.589 |
| ProbLog (full) | - | - | - | - | - | - |
| Prob. SQL (filtered) | 10.423 | 3.524 | 0.107 | 0.024 | 1.421 | 0.628 |
| Prob. SQL (full) | 8.846 | 5.488 | 0.097 | 0.011 | 0.888 | 0.617 |
| SQL (full) | 5.002 | 3.196 | 0.017 | 0.009 | 0.637 | 0.340 |

Table 4.7.: Query performance in seconds, averaged over 10 runs

rewriting step, the statement that Sandy graduated from State University with probability 1.0 is also included, resulting in her definitely being a person because of *graduatedfrom* having *person* as domain.

Tables 4.6 and 4.7 show the results of comparing *TIP-OWL* with ProbLog.

Table 4.7 shows the time needed for answering queries over the full dataset and the reduced one. To accommodate for the fact that ProbLog always has to load the data anew, loading times as shown in Table 4.7 have been subtracted from the query times for ProbLog.

As expected our approach takes significantly more time loading the data as index structures have to be created on disk and ProbLog only seems to do minimal preprocessing and keeps all the data in memory. The results in Table 4.7 show, however, that this effort is more than compensated by more efficient query answering.

The query response times clearly show that our database-driven approach is more efficient for handling large datasets. ProbLog is not able to answer any of the questions using the full dataset within a 30 minute timeout. Also for the filtered dataset, ProbLog fails for $Q_A$ and $Q_D$ with an out of memory error. For the queries where both return answers, our approach is between 15 and 30 times faster. We can observe that query processing even becomes more efficient for the larger dataset. After analyzing the generated query plans, we found that the query planner chooses a suboptimal query plan for the smaller dataset. We suppose this is due to weaker statistics. The overhead of the probabilistic SQL compared to the plain rewritten SQL seems to be proportional to the number of computed answers. $Q_A$, $Q_C$ and $Q_D$, and $Q_F$, with a larger number of results, are twice to five times as slow as the plain queries; $Q_B$ and $Q_E$ with very few results show almost no difference in query time.

| LUBM | Q1 | Q3 | Q4 | Q5 | Q6 | Q10 | Q11 | Q12 | Q13 | Q14 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | < 0.1 | < 0.1 | 1.5 | 2.7 | 0.7 | 0.3 | 0.2 | 0.8 | 0.3 | 0.3 |
| 10 | < 0.1 | 1.0 | 3.4 | 27.3 | 7.4 | 2.4 | 0.2 | 0.9 | 2.9 | 2.4 |
| 100 | < 0.1 | 32.2 | 50.5 | 350.9 | 74.2 | 33.2 | 0.3 | 2.0 | 76.2 | 33.2 |
| 200 | < 0.1 | 100.0 | 134.6 | 2 622.2 | 172.1 | 63.9 | 0.5 | 3.2 | 172.5 | 63.9 |
| 500 | < 0.1 | 201.8 | 440.2 | - | 508.0 | 192.0 | 1.1 | 6.9 | 612.1 | 1 941.9 |
| 1000 | < 0.1 | 643.8 | 904.7 | - | 874.7 | 365.1 | 1.6 | 232.3 | 1 430.3 | - |

Table 4.8.: Query response times (seconds) of our implementation on various sizes of the probabilistic LUBM dataset. A timeout (response time > 60 minutes) is denoted as "-".

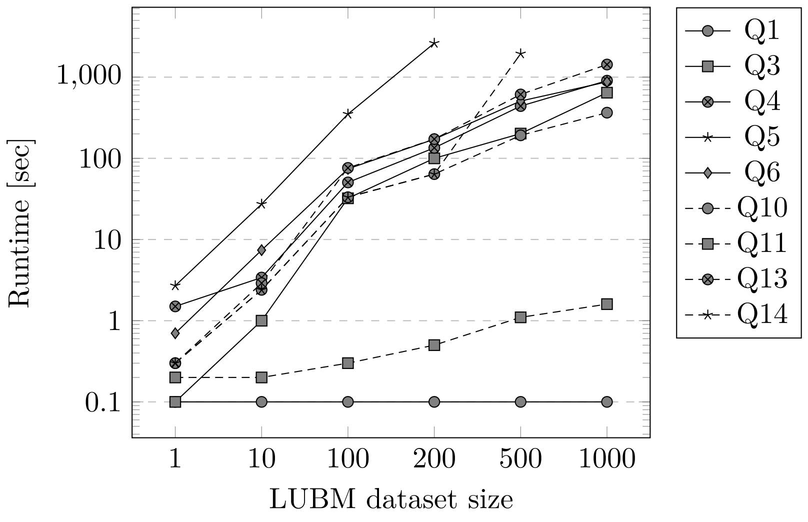


Figure 4.1.: Runtimes on datasets of various sizes

## LUBM Dataset

Table 4.8 and Figure 4.1 show the results using the probabilistic LUBM datasets. We only compared the performance of our implementation on different size of the data. We ran the queries with a timeout of 60 minutes. ProbLog is not able to handle even the smallest of those datasets.

The query response times show, that in general, the probabilistic reasoning does not have a negative impact on scalability. Overall, the times increase linearly in the size of the data. Query 1, which has a constant result that does not change with the size of the dataset, also has a constant response time. When processing Query 5, the database erroneously scans the complete table of data property assertions, which takes most of the time for computing results. This could be alleviated by tuning the query planner, resulting in a better query execution plan. Query 13 and especially Query 14 produce

| Dataset | LUBM 1 | LUBM 10 | LUBM 100 | LUBM 200 |
|---|---|---|---|---|
| QMpH | 418.6 | 66.4 | 5.4 | 0.2 |
| Optimum | 514.3 | 75.0 | 5.8 | 1.0 |
| % | 81.4% | 88.5% | 93.1% | 20% |

Table 4.9.: Query mixes per hour (QMpH) for different dataset sizes. The number indicates how often the set of benchmark queries could be executed within one hour. The queries are executed in random order.

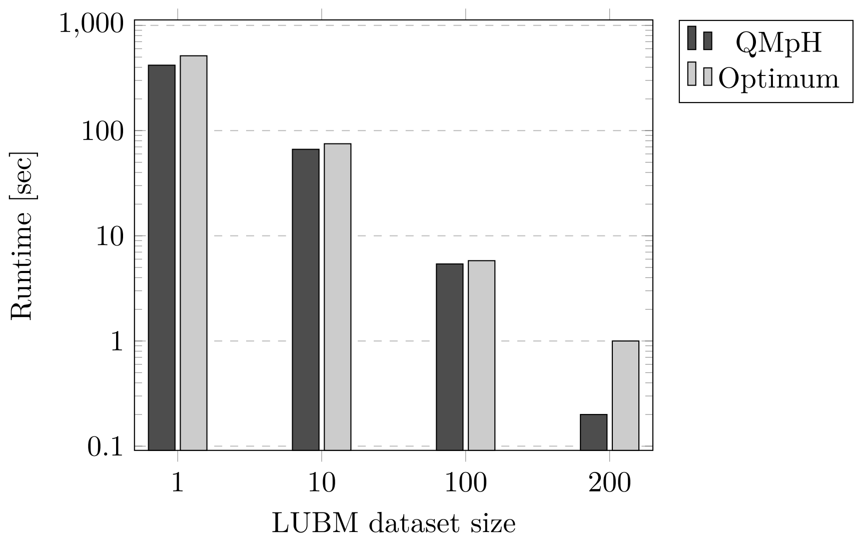


Figure 4.2.: Runtimes on datasets of various sizes

a large number of results, thus they become I/O-bound for larger datasets, i.e. their performance is limited by disk speed, resulting in a large jump in the query time for larger datasets. The storage for our test virtual machine are attached via network, resulting in this large drop in performance.

To evaluate the scalability under a more realistic workload we tested how often the set of all ten queries can be executed within one hour (inspired by the BSBM benchmark). The queries are executed in random order to counter caching effects. Table 4.9 shows the number of query mixes processed in an hour for different sizes of the LUBM dataset. Up to 70 million facts, the performance scales well and is close to the expected optimum based on the query times under ideal settings. For the smaller datasets, the response time are slightly farther away from the optimum due to a relatively higher overhead of establishing a database connection etc. Beyond 70 million facts, there is a huge drop in performance due to I/O-bound queries and the poor I/O performance of the virtual machine.

## 4.5. Related Work

We discuss three areas of related work:

1. Work which also investigates the use of SPARQL by real-world users.
2. Research that is focused on probabilistic query answering in general.
3. Work on probabilistic querying specifically in an OBDA setting.

### 4.5.1. Analysis of Real-World SPARQL Queries

Analysis of the characteristics of SPARQL queries in existing work is focused on deterministic queries. To the best of our knowledge, there is no analysis of the safeness of real-world SPARQL queries for probabilistic data.

Picalausa et al. analyzed 3 million queries from DBpedia query logs (Picalausa and Vansummeren, 2011). They come to the conclusion that the majority of the queries therein are tractable for the deterministic case.

Arias et al. (Gallego, Mario Arias Fernández, Javier D. Martínez-Prieto and de la Fuente, 2011) observe that most real-world SPARQL queries of the USEWOD2011 dataset (Berendt et al., 2011) containing queries from DBPedia and Semantic Web Dog Food) are star-shaped and do not contain property chains. As Jung et al. (Jung and Lutz, 2012) have shown that tractable queries are generally star-shaped, their findings also coincide with our results.

Han et al. (Han et al., 2016) also analyzed queries in LSQ. Similarly to Picalausa et al. they come to the conclusion that most queries are in PTime. They also show that most queries have a simple structure consisting of three or less triple patterns, making it less probable for them having a structure that is unsafe. An unsafe query must at least consist of either two triple patterns with a self-join (the same predicate), or three triple patterns (cf. unsafe queries given by Suciu et al. (Suciu et al., 2011)).

### 4.5.2. Probabilistic Querying

The two main areas of research in probabilistic querying are probabilistic programming and probabilistic databases (Theobald et al., 2013). Both have developed rather independently during the last decade but they share a common goal in making probabilistic inference scalable.

Probabilistic programming mainly focuses on learning statistical models from existing data. Its main concern is determining whether there exists a solution to a query rather than finding all solutions as probabilistic databases do. There are several formalisms, e.g. Markov logic networks (Richardson and Domingos, 2006), Bayesian networks (Jensen, 1996), or languages and tools like ProbLog (De Raedt et al., 2007), to describe probabilistic models. These formalisms and the tools using them, for example MarkoViews (Jha and Suciu, 2012) and BayesStore (Wang et al., 2008), can handle a large amount of uncertain data, however, none of them incorporates Semantic Web reasoning.

Similarly, probabilistic databases like Trio (Widom, 2004) or MayBMS (Huang et al., 2009) are designed for querying large probabilistic knowledge bases, but they are only handling relational data without rich semantic information. Probabilistic databases usually employ a *possible world semantics*: all possible values for uncertain values are represented at once. To enable efficient inferencing, they exploit independence assumptions, most commonly tuple-independence (all database tuples or rows are independent from each other) or block-independent (groups of tuples are independent from each other). The research on probabilistic databases draws from many years of experience with relational databases. From those, they adopt relational algebra – and thus SQL – or Datalog.

Theobald et al. (Theobald et al., 2013) give a more thorough overview of these two research areas, their methodologies, differences, and commonalities.

### 4.5.3. Probabilistic Ontology-Based Data Access

Apart from ProbLog, two other systems for probabilistic reasoning with a similar semantic are Pronto (Klinov and Parsia, 2013) and BUNDLE (Riguzzi et al., 2013). They can handle probabilistic knowledge bases formulated in $\mathcal{SROIQ}$ and $\mathcal{SHOIN}$(D), respectively. However, their main focus is not pOBDA, but probabilistic TBox reasoning (classification, satisfiability, …), thus their performance in query answering is very limited. Both can only run simple instance checking for single individuals and classes. Probabilistic deductive databases (Lakshmanan and Sadri, 1994) provide a similar solution, but to the best of our knowledge there is no system available and thus it is hard to estimate their scalability to large-scale knowledge bases.

Regarding the benchmark dataset, Klinov et al. (Klinov and Parsia, 2008) proposed a systematic approach to evaluate reasoning in probabilistic description logics which is, however, more geared towards complex TBoxes and not large-scale query answering. Lanti et al. (Lanti et al., 2015) very recently published a dataset, based on real world data, specially tailored for benchmarking OBDA systems. They also provide a generator to scale the dataset in size. It will be interesting to analyze their dataset and also extend it for benchmarking probabilistic OBDA systems.

## 4.6. Conclusion

In this chapter, first, we analyzed half a million queries from the LSQ dataset; those which can directly be translate to (union of) conjunctive queries. Checking for probabilistic query safeness we found nearly all queries to be safe and thus tractable when processed over probabilistic data. Relating to the whole set of queries, about half of them are safe; for the other half there is no information in our analysis. This shows that the general information needs a user of the public SPARQL endpoints serving as sources for the LSQ dataset has are also feasible to be fulfilled on uncertain data. Note that there is no uncertain version of those datasets. However, we argue that the users' information needs would not change over uncertain data, and that it is possible to create such a version of DBpedia, e.g. by incorporating information about trustworthiness in general or knowledge about how commonly a certain property is accurately extracted. To the best of our knowledge this is the first work that analyzes real-world SPARQL queries in the light of probabilistic data.

Second, we described a preliminary implementation of a probabilistic OBDA system for large-scale knowledge bases. It combines tractability for a certain class of queries with the benefits of ontology-based query rewriting. While making many simplifying assumptions the approach is well suited for large-scale knowledge bases with facts generated using machine learning techniques and provides a pragmatic alternative for theoretically more interesting but less feasible models as the ones proposed in BUNDLE (Riguzzi et al., 2013) and Pronto (Klinov and Parsia, 2013).

We used NELL and an adapted version of the LUBM benchmark to evaluate the system. NELL as a real world dataset gives valuable insight on the usefulness of probabilistic OBDA. It is, however, rather hard to scale to different sizes without generating datasets that contain no reasonable amount of answers. LUBM on the other hand is very easy to scale to various due to its synthetic data generator, but because of that it also lacks on complexity in the TBox. Jung et al. improved the TBox by extending the class hierarchy and adding existential restrictions but the ontology is still rather artificial and the queries are limited in their diversity.

The work presented in this chapter provides a thorough view on handling large-scale, uncertain data with Semantic Web technologies The results show that continuing research in that direction is both worthwhile and feasible in practice: Users can benefit from additional results from handling uncertainty and including reasoning, and the application in practice is feasible.

# 5 IT Risk Management in Large-scale IT Infrastructures

## 5.1. Introduction

IT risk management tries to find, analyze and reduce unacceptable risks in the IT infrastructure. Most commonly risk is defined as a set of triplets, each triplet consisting of a scenario, its probability and its potential impact (Kaplan and Garrick, 1981). In the IT environment these scenarios are typically called threats.

If a new threat surfaces, the IT risk management needs to asses its probability and evaluate its potential impact. Today's IT infrastructure has complex dependencies and a threat to a single component can threaten a whole network. Furthermore, single threats often have a very low probability but the combination of many threats can be a major risk to an IT infrastructure. Therefore, it is not enough to look at infrastructure components individually to determine the possible impact of a threat. While each year a huge number of new threats surfaces, old threats do not vanish (European Union Agency for Network and Information Security, 2013).

A fast response to a new threat is important to minimize the chance of exploitation. However, a manual threat analysis takes time. The complexity of today's IT infrastructure provides many indirect ways a single threat can affect different IT services and each must be analyzed. At the same time, the number of new threats is increasing, which leaves even less time to evaluate each new threat (IBM X-Force, 2013).

A semi-automated approach allows an easier handling of this complexity, but to our knowledge there exists none that incorporates dependencies and combinations of multiple threats. Even the tools to monitor and report risk are still in a premature state and there is a demand for tool supported risk assessment (Ernst & Young, 2013). While the measurement of IT infrastructure availability is already part of IT service management (Cabinet Office, 2011), it is only moderately used in IT risk management (Ernst & Young, 2013).

Root cause analysis (RCA) plays an important part in processes for problem solving in many different settings. Its purpose is to find the underlying source of the observed symptoms of a problem. IT plays an important role in processes in a wide area of business, thus a high availability and short response times to failures (e.g., failing e-mail deliveries, inaccessible websites, or unresponsive accounting systems) are crucial (IBM X-Force, 2013). Today's IT infrastructures are getting increasingly complex with diverse direct and transitive dependencies. This makes root cause analysis a time intensive task as the cause for a problem might be unclear or the most probable cause might not be the most obvious one. Therefore, automating the process of root cause analysis and helping an IT administrator to identify the source of a failure or outage as fast as possible is important to achieve a high service level (Ernst & Young, 2013).

In this chapter we present our approach to root cause analysis that uses Markov Logic Networks (MLN) and abductive reasoning to enable an engineer to drill down fast on the source of a problem. Markov Logic Networks provide a formalism that combines logical formulas (to describe dependencies) and probabilities (to express various possible risks) in a single representation. We focus on abductive reasoning in MLNs and show how it can be used for the purpose of root cause analysis. To our knowledge, the proposed approach is a novel method to root cause analysis that combines probabilistic and logical aspects in a well-founded framework.

Throughout this chapter, we illustrate our approach on a small case study. The IT infrastructure in our settings is comprised of a multifunction office printer that offers – amongst others – printing and scanning services via a network. These services use a mail and indirectly an LDAP service. Everything is dependent on the network and the power supply. This small case study already has dependencies that cross several different levels of infrastructure (services, server hardware, network hardware, power supply). We will expand this setting with possible causes for failure and probabilities for their occurrence. Theses risks are described in the *IT-Grundschutz Catalogues* by the German "Bundesamt für Sicherheit in der Informationstechnik" (Federal Office for Information Security) which is based on the ISO 27001 certification[1]. Furthermore, we evaluate the scalability of the approach on infrastructures generated randomly based on the structure observed in real-world environments.

Within our framework, the IT infrastructure is represented as a logical dependency network that includes various threats to its components. When a problem occurs, available observations are entered into the system which then generates the Markov Logic Network from the available observations, the given dependency network, and the general background knowledge related to the components of the infrastructure. Some of these observations might be specified manually, while other observations can be entered into the system automatically, e.g. via constantly running monitoring software. These observations are typically incomplete in the sense that not all relevant components are monitored, or not all problems are recognized. Thus, taking the given observations into account, there might still be a set of several explanations for the problem that occurred. Under the assumption that the modeled dependency network captures all

[1]`https://www.bsi.bund.de/EN/Topics/ITGrundschutz/itgrundschutz_node.html`

relations present in the infrastructure and all threats are adequately taken into account, the correct explanation will always be contained in that set.

We calculate, via abduction, the most probable cause for the current problem, which is then presented to the user, e.g., the administrator of the IT infrastructure. The user can then investigate if it is indeed the source of the problem. This might require to manually check the availability of some component or to analyze a log file. If the proposed explanation is correct, counter-measures can be introduced immediately. If the additional observations revealed that the calculated explanation is wrong, those new observations are entered into the system as additional evidence and a better explanation is computed. This iterative, dialog-based process is a practicable approach to quickly narrow down on a root cause. Ruling out certain causes automatically is desirable, however, it is obvious that most non-trivial problems still need to be checked manually by a user.

In our approach, we represent the given infrastructure and the possible risks as ontology. This allows us to automatically infer that certain threats are relevant for certain infrastructure components, or add logical constraints ensuring consistency. Relevant background knowledge can easily be maintained and used to generate the Markov Logic Network. Moreover, our approach can take into account known probabilities of risks and failures. These probabilities are derived from expert judgment or statistical data. Instead of computing multiple candidate explanations, which is possible in purely logic based approaches, we are able to generate the most probable explanation with our approach, while still leveraging the full power of an expressive, declarative framework.

This chapter is structured as follows. First, we present the theoretical underpinnings of our approach. In Section 5.2, we give a brief description to Markov Logic, introduce the general notion of abduction, and explain how abduction can be realized in the context of Markov Logic Networks. Furthermore, we give a short introduction to ontologies and their benefit in modeling IT infrastructures. In Section 5.3, we first present a typical scenario for root cause analysis. Then, we show how to model this scenario in our framework and describe how to apply abductive reasoning to find the most probable root cause. We present a workflow that illustrates how our approach is used in the context of a dialog-based process in Section 5.4. Furthermore, we conduct the evaluation of the scalability of the approach in Section 5.5. A tool we implemented to support the user in modeling the infrastructure and running a root cause analysis is presented in Section 5.6. In Section 5.8, we show how our approach is related to other works. Finally, we discuss the drawbacks and benefits of our approach in Section 5.9.

## 5.2. Preliminaries

This chapter also uses first-order logic and Markov logic networks as defined in Section 3.2 as foundation. In this section we introduce how abductive reasoning can be conducted in those formalism, and we detail how ontologies are used as basis for modeling an IT infrastructure.

The idea to use ontologies for modeling IT infrastructures has been proposed for several reasons. Vom Brocke at el. (vom Brocke et al., 2014) propose the use of ontologies to model the relationship between IT resources and business processes for the purpose of measuring the business value of IT. Ekelhart et al. (Ekelhart et al., 2006) provide a security ontology to support small and medium sized businesses IT-security risk analysis. Within our work we are distinguishing between acquisition, verification and maintenance on top of an ontological representation, and the probabilistic reasoning using Markov Logic Networks. This means that our work is compatible with the previously mentioned proposals, while we are able to conduct probabilistic reasoning required for root cause analysis. Thus, we can leverage the task-specific benefits of both formalisms.

### 5.2.1. Abduction in Markov Logic Networks

Abductive reasoning – or simply *abduction* – is inference to the best explanation. It is applicable to a wide array of fields in which explanations need to be found for given observations, for example plan or intent recognition, medical diagnosis, criminology, or, as in our approach, root cause analysis. According to (Kate and Mooney, 2009), abduction is usually defined as follows (Pople, 1973):

**Given:** Background knowledge $B$ and a set of observations $O$, both formulated in first-order logic with $O$ being restricted to ground formulae.

**Find:** A hypothesis $H$, also a set of logical formulae, such that $B \cup H$ is consistent and $B \cup H \vdash O$.

In other words, find a set of assumptions (a hypothesis) that is consistent with the background knowledge and, combined with it, explains the observation. It is the opposite of deductive reasoning which infers effects from cause.

The relation between root cause analysis and abductive reasoning is rather straightforward. In our approach, the background knowledge is the dependency network, respectively the Markov Logic Network to which we transform it. The dependency graph and Markov Logic Networks both are based on first-order logic as a formalism and thus conveniently are already in the desired logical representation. The observations, i.e., information about components being available or unavailable, are not part of the model but rather are directly provided as evidence to the MLN. We then try to prove through abduction that a specific threat – the most plausible cause – has occurred.

The inference mechanism in Markov Logic Networks is by default deductive, not abductive. Deductive reasoning draws new, logically sound conclusions from given statements. Kate et al. and Singla et al. (Kate and Mooney, 2009; Singla and Mooney, 2011) proposed methods – Pairwise Constraint (PC) and Hidden Cause (HC) model – that adapt Markov Logic Networks to automatically perform probabilistic abductive reasoning through its standard deductive reasoning mechanism. Their method augments the clauses of the MLN to support abductive reasoning as defined above. In general, the methods first introduce a reverse implication for every logical implication already present in the network. For example, if there are formulas $p_1 \rightarrow q$, ..., $p_n \rightarrow q$ in the MLN, the formula $q \rightarrow p_1 \vee \ldots \vee p_n$ is added to the MLN.

In a second step the model is extended with mutual exclusivity constraints that bias the inference against choosing multiple explanations. The reverse implications and the mutual exclusivity clauses are modeled as soft rules and may occasionally be violated, for example, if multiple explanations provide a better proof for the hypothetical root cause than a single explanation. We follow this basic idea, however, we argue that the mutual exclusivity constraints are not required in the application that we are interested in.

Revisiting the example above, we have to add the following reverse implication to conduct abductive reasoning:

$$\text{hasHobby}(y, z) \rightarrow \text{friends}(x, y) \wedge \text{hasHobby}(x, z) \tag{5.1}$$

This implications ensures that any additional grounding from hasHobby($y$, $z$) has some corresponding atoms friends($x$, $y$) and hasHobby($x$, $z$) or is forbidden otherwise.

## 5.3. Root Cause Analysis with Markov Logic Networks

Root cause analysis is the task of finding the underlying cause of an event. It is often applied to analyze system failures. System failures are commonly caused by a cascade of events. The goal of a root cause analysis is finding the original reason for the failure, so that a sustainable solution can be provided (Rooney and Heuvel, 2004). Root cause analysis typically comprises two phases: the detection of an event and the diagnosis of the event. In our work, we are concerned with the second phase and assume that a failure has already been detected.

In this section, we first illustrate the infrastructure of our case study. Then we show how to model dependencies and risks as a set of first-order formulas. While this model is the core component for computing the MAP state, which corresponds to the root cause, we also leverage an ontological model to describe and maintain the infrastructure, which is then used to automatically construct some of the relevant dependency and risk assertions as first-order formulas. Then, we explain how we implemented abduction in our Markov Logic Network and show special properties of our settings which simplify the general approach of abductive reasoning. Finally, we explain how the method is integrated in an iterative process.

### 5.3.1. Scenario Setting

In the subsequent sections, we discuss our approach with the help of an infrastructure shown partially in Figure 5.1. This small sample revolving around an office multifunction printer consists of the following components:

- The basic dependency for all components is the *Power Supply*. The only risk that can affect it is a general outage.
- The *Network Switch* connects the other components. It only depends on the power supply; it has multiple risks, e.g. congestion, overheating, or denial-of-service attack, not explicitly depicted Figure 5.1.
- The two servers *mail.uni-ma* and *cas.uni-ma* each offer one service, i.e. the *Mail Service* and an *LDAP authentication service*. The Mail Service uses the LDAP service to authenticate users. Both servers are subject to various threats, e.g. malicious software, DOS attacks, overloading, or compromise of the system.
- The *Office Printer* offers three services: *Copying*, *Printing*, and *Scanning*. It also has various problem sources, e.g. lack of resources or a technical malfunction.

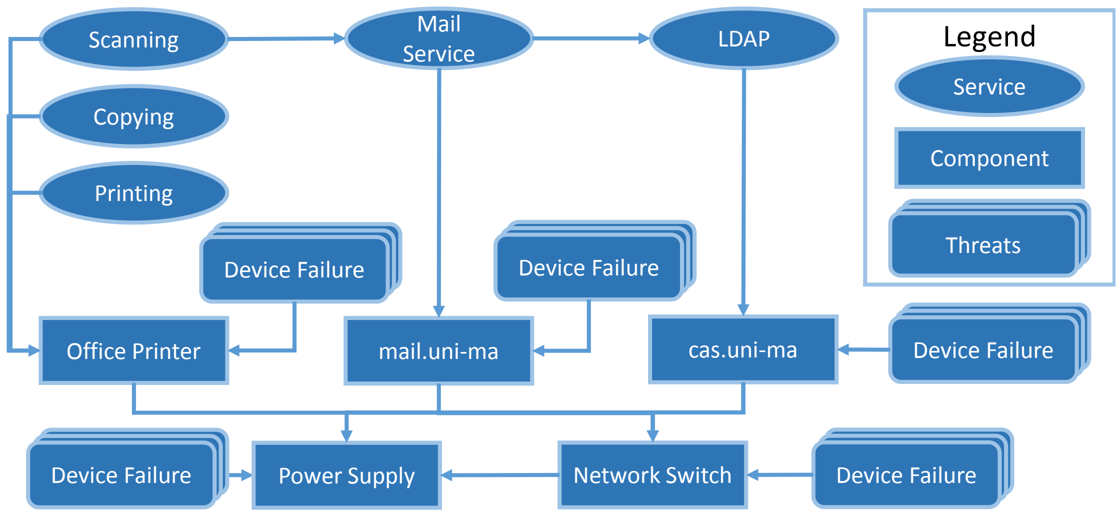


Figure 5.1.: Case Study: Office multifunction printer with multiple risks/threats attached (for brevity risks are grouped as *Device Failure*). In this small example we do not consider redundant components, i.e. all edges represent *specificallyDependsOn* relations.

The threats we are using in our example are defined in the *IT-Grundschutz Catalogues* (Bundesamt für Sicherheit in der Informationstechnik, 2016, p. 417ff.):

- *Disruption of power supply*: Short disruption of the power supply, more than 10 ms, or voltage spikes can damage IT devices or produce failures in its operation.

- *Failure of Devices or Systems*: No equipment runs infinitely and a hardware failure in an IT device will happen if it runs long enough. Beyond the damage of the device, the downtime has an effect on the processes that depend on the device or can even damage other devices, e.g. in the case of a cooling system.
- *Systematic trying-out of passwords*: An attacker can gain access to a system by discovering the password of the system through systematic trial-and-error.
- *Lack of Resources*: If the given resources (for example bandwidth, disk space or personnel) in an area of the operation are smaller than the current demand, a bottleneck occurs. This results in congestion and failure of operation.
- *Malicious software*: Malicious software tries to execute a process that is unwanted or damaging for the owner of the device that runs the software. This includes viruses, worms and Trojan horses.
- *Misuse of spanning tree*: An attacker can use Bridge Protocol Data Units (BPDUs) to initialize the recalculation of the switch topology. This can be used to disrupt the availability of the network.

The *IT-Grundschutz Catalogues* are a comprehensive collection of threats and safeguards for various parts of an IT infrastructure[2]. They are created and maintained by the German Federal Office for Information Security[3], and compatible to the ISO 27001 certification[4].

### 5.3.2. Modeling Dependencies and Risks

The foundation of our root cause analysis is the dependency model. It uses first-order logic to describe various aspects of the IT infrastructure. Our basic model uses five predicates:

- **specificallyDependsOn(*x*,*y*)** specifies that component $x$ is specifically dependent on component $y$, e.g. the mail service that runs on the mail server. This predicate does not allow for any redundancy of $y$.
- **genericallyDependsOn(*x*,*y*)** specifies that component $x$ depends on $y$. $y$ may be replaced by some other redundant component. An example is a server running on the normal power supply vs. some uninterruptible power source (UPS).
- **redundancy(*x*,*y*)** states that $x$ and $y$ are redundant, i.e. they offer the same services and can replace each other in the case of failure.
- **affectedByRisk(*x*,*y*)** assigns the risk $y$ to component $x$, i.e. $y$ is a threat that endangers the functionality of a component and it can affect $x$.

[2]`https://www.bsi.bund.de/EN/Topics/ITGrundschutz/itgrundschutz_node.html`
[3]Bundesamt für Sicherheit in der Informationstechnik (BSI)
[4]`http://www.iso.org/iso/home/standards/management-standards/iso27001.htm`

- **unavailable($x$)** designates a component $x$ as unavailable, e.g. offline or not functioning properly.

The distinction between dependencies that allow for redundancy and those that do not helps to improve reasoning performance. For specific dependencies, no further checks for redundancies not to be performed in case of a failure and all dependent components are directly set to be unavailable.

The shown predicates are only one way of modeling the infrastructure; other predicates are possible. For example, a simple modification are additional types of dependencies that distinguish between, e.g., power and network connections, or technical and human errors. This allows for additional constraints to ensure the consistency of the model, for example by requiring that each component has at least one power and one network connection. This change can directly be made and later checked in the ontology that specifies the dependency graph. The subsequent translation to MLN does not need to be adapted. Another, more complex example is to model relationships as individual nodes, which enables the user to specify more details about it, e.g., the probability of a broken network cable. This also requires changes in the rules of the MLN to account for the additional relationship nodes. However this only needs to be done once, when deciding to model the infrastructure in this way. It is possible to mix both modeling approaches, for example to include detailed information about relationships where available, and ease modeling for the user where it is not. We chose the predicates as presented for reasons of brevity and easier understanding.

Formulae 5.2a to 5.2f depict the basic MLN program built from those predicates:

$$\begin{aligned}\langle \forall x, y \ (\mathrm{unavailable}(y) \wedge \mathrm{specificallyDependsOn}(x, y) \\ \Rightarrow \mathrm{unavailable}(x)), \infty\rangle\end{aligned} \tag{5.2a}$$

$$\begin{aligned}\langle \forall x, y \ (\mathrm{unavailable}(y) \wedge \mathrm{genericallyDependsOn}(x, y) \\ \wedge \neg\exists z \ (\mathrm{redundancy}(y, z) \wedge \neg\mathrm{unavailable}(z)) \\ \Rightarrow \mathrm{unavailable}(x)), \infty\rangle\end{aligned} \tag{5.2b}$$

$$\langle \forall x, y \ (\mathrm{redundancy}(x, y) \Rightarrow \mathrm{redundancy}(x, y)), \infty\rangle \tag{5.2c}$$

$$\begin{aligned}\langle \forall x, y \ (\mathrm{redundancy}(x, y) \wedge \mathrm{redundancy}(y, z) \\ \Rightarrow \mathrm{redundancy}(x, z)), \infty\rangle\end{aligned} \tag{5.2d}$$

$$\langle \forall x, y \ (\mathrm{affectedByRisk}(x, y) \Rightarrow \mathrm{unavailable}(x)), \infty\rangle \tag{5.2e}$$

$$\begin{aligned}\langle \forall x, y \ \neg(\mathrm{specificallyDependsOn}(x, y) \\ \wedge \mathrm{genericallyDependsOn}(x, y)), \infty\rangle\end{aligned} \tag{5.2f}$$

Formula 5.2a forbids any world where infrastructure component $y$ is unavailable and infrastructure component $x$ is available, if there is a specific dependency from $x$ to $y$. Formula 5.2b is similar to Formula 5.2a, but phrased for generic dependencies with redundancies. Provided $x$ is generically dependent on $y$ and $y$ is unavailable, then $x$ is unavailable only if there is no other component $z$ that is redundant to $y$ and available. Thus, a component is only available if every specific dependency is available or if at least one redundant component is available for each generic dependency, respectively. The symmetry and transitivity of *redundancy* is modeled by Formulae 5.2c and 5.2d. By adding these two formulas, we ensure that it is not required to specify redundancy for all pairs in both directions. If we extend an infrastructure with an additional redundant component, we only need to add a single statement instead of specifying the information for all pairs in the group of redundant components. Formula 5.2e enforces that a component $x$ that is affected by the effects of a risk $y$ becomes unavailable. The predicates *specificallyDependsOn*$(x, y)$ and *genericallyDependsOn*$(x, y)$ are mutually exclusive (Formula 5.2f).

The known dependencies, risks, and unavailabilities are modeled as evidence as shown below. Note that these formulas are only three examples for all formulas required to describe the infrastructure depicted in Figure 5.1.

$$\langle \text{specificallyDependsOn}(\textit{MailService}, \textit{mail.uni-ma}), \infty \rangle \tag{5.3a}$$

$$\langle \text{affectedByRisk}(\textit{mail.uni-ma}, \textit{MaliciousSoftware}), -1.2 \rangle \tag{5.3b}$$

$$\langle \text{affectedByRisk}(\textit{mail.uni-ma}, \textit{DDOS}), -3.2 \rangle \tag{5.3c}$$

Formula 5.3a is a hard fact, which states that the *MailService* depends on the server *mail.uni-ma*. The soft Formula 5.3b encodes that *mail.uni-ma* can be affected by *MaliciousSoftware*. This formula has a negative weight, i.e. it translates to a low probability. *mail.uni-ma* also has *DDOS* as a second risk (Formula 5.3c). Generally, there is no upper limit to the number of risks that can be attached to a component

As described before, the dependency relation must hold in every possible world. The soft formula, however, is not fulfilled in most of the worlds due to the negative weight. In fact, if only this evidence is given, the most probable world does not include it, as it lowers the sum of the weights of all formulas.

Achieving high availability, defined as up to 5 minutes unavailability per year, is a longstanding goal in the IT industry (Gray and Siewiorek, 1991). Continuous monitoring of availability is part of IT service management best practices like the Information Technology Infrastructure Library (ITIL) (Cabinet Office, 2011). We define availability as the probability that a system is reachable and working properly. The availability of a system can be determined as follows:

$$Availability = \frac{Uptime}{Uptime + Downtime} \tag{5.4}$$

Unavailability is the inverse of availability:

$$Unavailability = 1 - Availability \tag{5.5}$$

The weights for new threats need to be estimated. We propose involving a domain expert in the estimation of how much the availability of the directly affected infrastructure component should be reduced. This allows us to learn a weight for the new threat and to determine how the new threat indirectly affects the availability of other components.

Determining the correct weight for the evidence is not trivial (Jain et al., 2007). However, there exist efficient learning algorithms for MLNs (Richardson and Domingos, 2006). Those algorithms can either work on collected data, for example from monitoring systems that provide uptime and downtime statistics, or based on estimations from domain experts or vendor specifications. Additionally, approximating the correct weights is sufficient in our use case, as we are not interested in the absolute probability of a specific root cause occurring, but just which root cause is most likely given some evidence.

### 5.3.3. Infrastructure Components and Background Knowledge

Our basic dependency model contains relatively simple first-order rules. The dependencies within this basic model are ignorant with respect to the types of the entities that are linked. However, we know that an IT infrastructure is typically a network built from different types of entities. In particular, the dependencies between these entities are restricted with respect to their types. We know, for example, that each server must depend on a power supply, while it makes no sense to have an explicit dependency between a service and a power supply. This dependency is indirectly modeled by the fact that a service must run on a server that depends on a power supply. In our approach, we propose the use of a Description Logics (DL) ontology to model the types of and the relationships between the components of the IT infrastructure. The formulas of a DL ontology are divided into TBox axioms and ABox assertions (see also (Baader et al., 2003)). The examples given above will be encoded as axioms of the TBox. A TBox contains terminological axioms that describe the relations and types that are used (later) in the ABox to make concrete assertions.

With respect to the ABox, we have developed a graphical user interface to specify and visualize the concrete infrastructure. It is used to add components to the model of the infrastructure that are typed in terms of the TBox vocabulary. The user interface, presented in more details in 5.6, is also used to specify the observations and to compute a root cause whenever a root cause analysis is required. In order to define the TBox, we have used the ontology editor Protégé to model the TBox axioms (Musen, 2015). Protégé helps to abstract from the concrete encoding of the axioms and supports views that are also common to users that have only a limited experience in logical modeling.

We use the TBox to distinguish between the different types of infrastructure components, e.g., `Service`, `Server`, `Switch`, or `PowerSupply`. For these types we add axioms to specify both required and impossible dependencies. For example, we enforce that each service depends (specifically or generically) on a server, while we do not allow a direct dependency between service and power supply. These constraints can be used to check the consistency of an ABox that uses these axioms. Our user interface can use these reasoning services on the fly to check after each modification whether the resulting ontology is still consistent. This helps to detect both errors and missing dependencies during the knowledge engineering process.

Furthermore, we can use the TBox to specify concrete subtypes for each of the main types. An example might be the distinction between *FlashMemory*, *OpticalDisc*, *MagneticDisk*, and *MagneticTape* as sub types of *StorageComponent*. *FlashMemory* can again be divided into *FlashDrive*, *MemoryCard* and *SolidStateDrive*, for example. Note that such a fine-grained distinction is not required by our approach. The basic dependency model and our tool for defining the infrastructure works already with very basic types. In the simplest case we specify only one type called *Component*. However, a fine-grained typology has several advantages. We mentioned already the reasoning capabilities in the paragraph above. Another advantage is the specification of type specific risks and their generic probabilities. For example, we can add information about the failure rate (in the form of a weight) of a specific hard drive model as background knowledge as follows:

$$\langle(\text{SCSIHardDrive}(x) \sqsubseteq \exists\text{affectedByRisk.HeadCrash}), 0.0015\rangle \tag{5.6}$$

The hard drive model *SCSIHardDrive* is described as hard drive that has a certain risk of a head crash. The probability attached to this formula might have been derived from available failure rates. Note that we can specify directly a probability that will be translated to the corresponding weight in Markov Logic. If required, we can also use the ontology to add further types related to, e.g., the manufacturer of the drive, since it might be known that drives produced by a certain company have a lower failure rate. By defining a concrete drive as instance of this type, it inherits all the properties of this type, i.e. the weighted risk of a head crash. This helps the knowledge engineer to define the components of an infrastructure without explicitly specifying each risk and its probability explicitly.

The ontological representation is automatically translated to the first-order Markov Logic formalization on the fly whenever a root cause analysis is computed. Niepert et al. (Niepert et al., 2011b) have shown that such translation is possible in general. For our purpose we have chosen a similar approach, however, type assertions and TBox axioms are not directly translated but are taken into account when associating risks with their weights to concrete components of the infrastructure. Thus, we leverage the ontological representation both for automatically detecting inconsistencies when modeling the infrastructure and for assigning weighted risks to types, which results in the assignment of generic risk weights if no specific information, e.g., from the history of the component stored in log-files, is available.

### 5.3.4. Computing Explanations

We now detail our approach and describe how the Markov Logic Network is constructed and extended, and how we use abductive reasoning for root cause analysis. The construction from background knowledge and extension for abduction of the Markov Logic Network is only done once and does not have to be changed during the root cause analysis. According to the method proposed in (Kate and Mooney, 2009) we have to add one reverse implication for the Formulae 5.2a, 5.2b, and 5.2e:

$$\begin{aligned}
&\forall x\ (\text{unavailable}(x) \\
&\Rightarrow (\exists y\ (\text{specificallyDependsOn}(x, y) \wedge \text{unavailable}(y))) \vee \\
&\quad (\exists y\ (\text{genericallyDependsOn}(x, y) \wedge \text{unavailable}(y) \\
&\quad\ \wedge\ \neg\exists z\ (\text{redundancy}(y, z) \wedge \neg\text{unavailable}(z)))) \vee \\
&\quad (\exists y\ (\text{affectedByRisk}(x, y)))
\end{aligned} \tag{5.7}$$

Additionally, Kate et al.s' method requires clauses for mutual exclusivity to be added. The purpose of these clauses is to *"explain away"* multiple causes for an observation and prefer a single one (Pearl, 1988).

$$\begin{aligned}
\forall x\ (\text{unavailable}(x) \Rightarrow\ & [\neg\exists y(\text{specificallyDependsOn}(x, y) \wedge \text{unavailable}(y))] \vee \\
& [\neg\exists y, z\ (\text{genericallyDependsOn}(x, y) \wedge \text{redundancy}(y, z) \wedge \\
& \text{unavailable}(y) \wedge \text{unavailable}(z))])
\end{aligned} \tag{5.8a}$$

$$\begin{aligned}
\forall x\ (\text{unavailable}(x) \Rightarrow\ & [\neg\exists y(\text{specificallyDependsOn}(x, y) \wedge \text{unavailable}(y))] \vee \\
& [\neg\exists y\ \text{affectedByRisk}(x, y)])
\end{aligned} \tag{5.8b}$$

$$\begin{aligned}
\forall x\ (\text{unavailable}(x) \Rightarrow\ & [\neg\exists y, z\ \text{genericallyDependsOn}(x, y) \wedge \text{redundancy}(y, z) \wedge \\
& \text{unavailable}(y) \wedge \text{unavailable}(z)] \vee \\
& [\neg\exists y\ \text{affectedByRisk}(x, y)])
\end{aligned} \tag{5.8c}$$

The reverse implications (5.7) as well as the mutual exclusivity clauses (5.8) are usually modeled as soft clauses. In general, for each set of reverse implications $P_i$ with the same left-hand side, $(\frac{|P_i|^2+|P_i|}{2}) \in O(n^2)$ mutual exclusivity clauses are added.

However, different from networks in that general method, our approach exhibits a property that simplifies the additional rules needed for abduction: All the weights in the evidence are negative – based on the reasonable assumption that threats and risks only occur rarely, i.e. components are available more than 50% of the time. This property allows us to reduce the size of the Markov Logic Network by leaving out the mutual exclusivity clauses completely: Due to the reverse implication, the MLN solver has to chose one cause to make the clause *true*. However, as all causes have negative weights and thus every cause set to true is lowering the sum of the weights of a possible world, the solver is already biased against choosing multiple explanations. This saves us from generating the quadratic number of mutual exclusivity clauses.

After constructing and extending the Markov Logic Network, we can conduct the root cause analysis. The overall process flow of our approach is depicted in Figure 5.2. The analysis is a dialog-based and iterative process, with interaction between our system and an administrative user. A fully automatic workflow is desirable, however, not every information can be retrieved directly and sometimes manual investigation of log files or on the status of components is necessary.

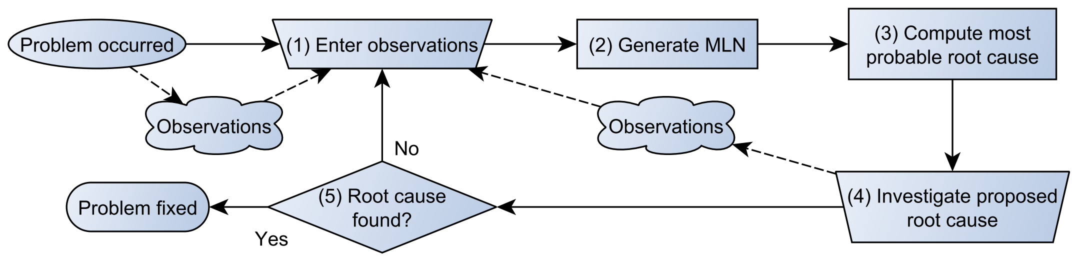


Figure 5.2.: Process flow for our approach on root cause analysis. Rectangles denote automatic action. Trapezoids require manual interaction by an administrative user. Clouds represents observations made and entered by a user.

In its normal state, without any hard evidence about availabilities or unavailabilities, all components are assumed to be available. Thus, when calculating the MAP state, it contains all components as available. When a problem occurs the user is required to provide observations as evidence for the MLN (Fig. 5.2: Step 1). These observations include any certain information about available and unavailable components. At least one unavailability must be specified to run the root cause analysis, however, providing more information is possible and will increase the accuracy of the analysis. For example, the user can enter that printing (over the network) is not possible, although the network is functional as browsing the internet still works. This results in hard evidence for the printing service being unavailable and network services and hardware required for internet access being available. The presented model only supports hard evidence about available and unavailable components. It is possible to extend this to also allow soft evidence – e.g. when availabilities are checked automatically and there is some probability for error – handled similarly to information about threats. However, this will also have some impact on the performance of the approach, as unavailabilities are not just promoted through the dependency graph, but also influence the weights when calculating the MAP state.

Our approach extends the Markov Logic Network with the new evidence (Fig. 5.2: Step 2) and uses an MLN solver to run MAP inference on it (Fig. 5.2: Step 3). The calculated MAP state contains the evidence provided by the user (this must be always fulfilled), components being unavailable due to a direct or indirect dependency on components observed as not available, and (at least) one root cause that explains the unavailabilities. Components which are not affected by specified observations or the calculated root cause are listed as available.

The root cause fulfills the following properties:

- It explains all unavailabilities in the evidence. This is the case due to the additional reverse implications.
- It is not affecting any component stated as available in the evidence. Otherwise a hard rule would be violated.
- It is the most probable cause for all the observations given as evidence and the risk probabilities specified as weights.

We make the assumption that all causes are unlikely (they appear less than 50% of the time). Thus, their weights are negative. As the objective of the MAP state is maximizing the sum of all weights, only the most likely cause that explains all observations is included. A less likely cause has a higher negative weight, causing the sum of the weights to be lower than optimal, and thus getting rejected.

Note that due to the soft formulas used for abduction, our approach only encourages to calculate a single root cause, but does not enforce it. It only presents multiple possible root causes, if the sum of their weights is less than the weight of a single possible cause. If there are two possible root causes with the same weight, only one is presented at random.

The user then has to investigate the presented root cause (Fig. 5.2: Step 4). If it is the source of the observed problem, the analysis is finished and the cause can be fixed. Otherwise the process starts over from the start where the user enters additional observations (Fig. 5.2: Step 5). Those new observations can either be gathered while investigating the proposed root cause, or, for example, the user can verify the state of components that should also be affected by this cause.

### 5.3.5. Limitations

Limiting factors for our approach is the expert knowledge required to build the background knowledge, and the worst-case performance of inference. Modeling the background knowledge requires good domain knowledge and experience in modeling ontologies. Furthermore, the model needs to be sufficiently complete to gain valuable results from the root cause analysis, as the approach is very fragile in the case of modeling errors. For example, if some critical, yet subtle, dependency is missed, the analysis might never provide any meaningful results. Many modeling errors can be checked with logical rules (e.g. every hardware component must be connected to a power source), but paying attention to capturing the infrastructure correctly and completely is of high importance. This factor is less relevant in companies that already use semantic technologies for IT infrastructure management, for example as presented in (Chen et al., 2013; Hess et al., 2010). This data can directly be used, mostly likely with only minor adjustments.

Another limitation is the worst-case complexity of MAP inference, which is NP-complete. However, there are efficient approximation algorithms for MAP inference (cf. (Kimmig et al., 2014) for a survey on different approaches), and as shown in Section 5.5 this is rarely an issue in realistic infrastructures.

## 5.4. Exemplary Scenarios

The following section describes the application of our approach to two different scenarios. Those scenarios represent two incidents that occurred in our infrastructure, which we then analyzed in hindsight with the presented approach.

### 5.4.1. Scenario Analysis

The following two scenarios illustrate failures that occurred in our IT infrastructure during the last months. Together with our system administrators, we modeled our infrastructure, analyzed these scenarios in hindsight, and tested the usefulness of our approach in retrospective. We used RockIt (Noessner et al., 2013), a highly optimized and scalable MLN solver, to compute the MAP state.

The first scenario is the one depicted in Figure 5.1, revolving around the malfunction of our office multifunction printer. The printer offers three services: copying, printing via the network, and scanning to PDF which is then sent to an email address. A user reported the printer being broken, as scanning to PDF no longer worked. To check the proper functioning of the device, the administrator sent a print job and did a photocopy. Both tests worked successfully. Sending a test mail from his own account, the administrator also found the mail service working correctly. Further investigation finally revealed that the root cause of the scanning problem was a suspension of the account the printer used for the LDAP authentication. However, this cause was only considered after several discussions with two expert administrators involved.

We applied our approach to this scenario. The MLN was constructed automatically from the background knowledge that we maintained as a set of first-order formulas. We enter the observations *available(Printing)*, *available(Copying)*, and ¬*available(Scanning)* and computed the most probable root cause. The MAP state that was generated as solution contained the root cause *affectedByRisk(cas.uni-ma, Systematic trying-out of passwords)*. While we could not definitely decide, in retrospective, if this risk was the underlying reason for the failure of the server *cas.uni-ma*, an authentication problem related to *cas.uni-ma* was definitely the cause for the problem.

The second scenario is an outage of our internal Subversion server. It involves more components than the previous scenario and benefits from the iterative approach. The Subversion server is hosted on a virtual machine that is running on a blade server. Subversion was responding slowly and took long time for many operations. Neither Subversion nor other processes on the virtual machine showed considerable resource utilization. Investigating resource usage on the blade server first did not reveal any abnormality. Later, a user discovered that our external website behaved similarly in performance as the SVN. This observation was first attributed to a slow internet connection in general, but we then discovered that the web server, which was hosted in a different VM but on the same blade server, produced very high network traffic, starving all other services. A member of our group had released a data set of several gigabytes in size, that was downloaded a few hundred times concurrently. That lead to congestion on the network interface of the server. Moving the download to another physical server resolved the problem and the behavior of the Subversion server and our website went back to normal.

Analyzing this scenario with our approach, first, we only entered the observation of the unavailability of the SVN service: ¬*available(Service_SVN)*. The computed MAP state proposed *affectedByRisk(VM_Subversion, Overload)* as root cause. Making the observations *available(VM_Subversion)* and ¬*available(Service_WebHosting)*, however, rules out this cause. The next iteration offered *affectedByRisk(NetworkInterface_BladeServer, Congestion)* as root cause. This risk has a high probability for that server which is running various other virtual machines, all hosting services sensitive to a high network load. The lack of other resources, e.g. CPU or RAM, is modeled as less probable, because all those services are usually not very computational complex or requiring lots of memory. For this scenario, our approach proposed reasonable root causes which we retrospectively could verify as the reason for the outage. The manual handling of the incident involved more guesswork by the system administrators and was long winded.

## 5.5. Evaluation of Scalability

We evaluated the scalability of the approach by automatically generating infrastructures of different sizes. The data generator was carefully modeled to inherit properties observed in real-world scenarios. We then simulated different numbers of root causes and observed offline components. We applied our approach to compute the root causes and measured the runtime.

### 5.5.1. Data Generation

As manually modeling infrastructures of various sizes to evaluate the scalability of the presented approach is not feasible – due to data not being publicly available and the effort required in modeling – we implemented a data generator that can create random infrastructures of any size. To create models which are close to real-world scenarios, we also discussed general properties of IT infrastructures with the experts when recording the scenarios described above, and include this knowledge in the design of the generator. We made the following observations which are reflected in the implementation:

- The dependency graph of an IT infrastructure has usually a limited depth: the basic layers of an infrastructure are power sources, network components, servers, and services.
  Power sources are located at the bottom layer without any dependencies to other components.
- Network components could technically be layered very deep, however instead of cascading several small switches and routers to connect many devices, generally a bigger switch is used, e.g. resulting in  16 million possible devices with a cascade of 64-port switches of depth 4.
- In our model servers rely on power sources and switches. Each server can offer services that then can depend on each other.
- Dependencies between services are naturally the most complex of the whole infrastructure. They can be layered over multiple services, e.g. starting from a low-level SAN service (storage area network) providing storage, over a virtualization using the SAN, a virtual server offering for example an LDAP service to the mail server using that for authentication. The dependency chain can become slightly longer by having another service providing email notifications and probably one or to more services offering even higher level operations, but in reality there is a limit to that depth. This complex example has a depth of 13; to provide some room for even more complex scenarios we imposed a limit of 16 to the maximum depth for the infrastructure the generator can create.
- Another observation that already becomes obvious in this small example, is that there are some services that are very central, like the storage (SAN) or authentication via LDAP which are used by almost all other services, and services that are used by few others, e.g. printing, which might only be used by persons (which are not part of the model). We factored this into the generation by drawing the possible dependencies from a normal distribution. For example, following from the *three-sigma rule of thumb* that nearly all values are within three standard deviations of the mean ($Pr(\mu - 3\sigma \leq x \leq \mu + 3\sigma) \approx 0.9973$), we draw from a normal distribution with $\mu = 500$ and $\sigma = \frac{500}{3}$ for an infrastructure of size 1000, or more general: for an infrastructure of size $n$ we have $\mu = \frac{n}{2}$ and $\sigma = \frac{n}{6} \equiv \frac{\mu}{3}$.

Additionally, the following constraints must hold:

1. Every network component and server is connected to exactly one power source.
2. Every network component and server is connected to one other network component or the root network component (e.g. representing some central switch or external internet connection).
3. Every service is running on exactly one server.
4. Every server is running some service (a server without offering a service is of no use).

5. A switch has $\approx$ 32–64 connections.
6. A server offers $\approx$ 1–3 service.
7. A service directly depends on $\approx$ 1–3 other services.
8. Every component has $\approx$ 1–6 risks attached to it.
9. Risks have a probability $0 < p < 0.5$, following a normal distribution with $\mu = 0.1$ and $\sigma = 0.05$.

Offline components are then defined by first choosing one or two root causes and determining all of their dependent components. Then for different datasets we picked one to five of those and marked them as offline in the model. This way calculating the root cause is more similar to real-world scenarios, where the component observed as offline is often not the real cause of the problem.

### 5.5.2. Scalability Results

We ran the approach 10 times for each size of the dataset and different amounts of errors. The average runtime and its standard deviation are shown in Table 5.1. Having different numbers of root causes and observed offline components had no significant impact in the runtime, thus we omitted those datapoints and only report the numbers aggregated by size.

| Dataset Size | Avg (sec) | StDev (sec) |
|---:|---:|---:|
| 1000 | 1.13 | 0.10 |
| 10 000 | 8.89 | 1.40 |
| 100 000 | 112.58 | 15.43 |

Table 5.1.: Average runtimes on infrastructures of various sizes

The results for different sizes of infrastructures are not calculated in real-time. However, two minutes for the largest dataset is still a reasonable amount of time to wait for results.

More importantly, the scalability of the approach is linear in the number of components which is a very favorable result, given the complexity of the calculation, as is apparent in Figure 5.3. This is mainly explained by the relatively simple structure of dependency graphs for IT infrastructures which are mostly tree-like, e.g. without circular dependencies, and do not exhibit the complex patterns for which MAP inference is NP-complete.

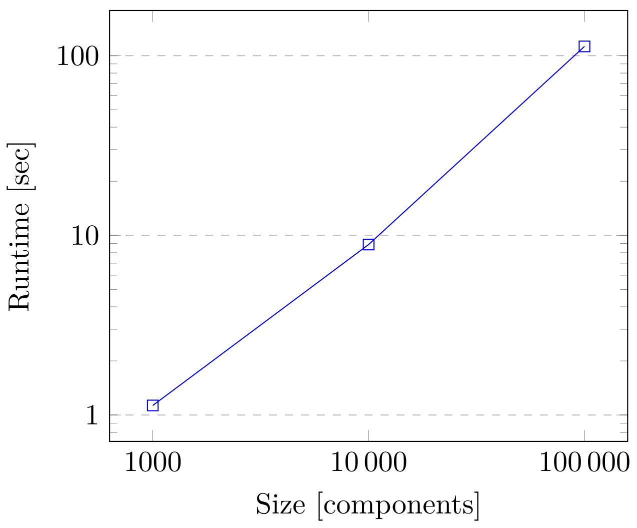


Figure 5.3.: Runtimes on infrastructures of various sizes

## 5.6. Tool Support

We implemented our approach in an open-source tool called RoCA. A screenshot of the user interface showing the clipping of a small dependency graph and the observations provided by the user is presented in Figure 5.4. It shows a *CMS service* that is running on two redundant *Apache web servers*, the *Mail service* running on some *New server*, and a *Printer*. All of those depend on some not shown network connection and power source. A user provided evidence about the CMS not working (depicted in red), but the printer being functional (green) – for the other components the status is unknown. The graphical user interface vastly increases the usability of our method, as a user does not need knowledge about logical first-order formulas, Markov Logic Networks, or ontologies for entering evidence or running a root cause analysis. The tool and its source code are freely available for download: https://github.com/dwslab/RoCA.

### 5.6.1. Required Data

The required data to run a root cause analysis is the background knowledge and the dependency graph with evidence about available and unavailable components.

RoCA uses the ontological representation that we described above as background knowledge. Usually, the ontology is defined once and represents the vocabulary used to describe the IT infrastructure. Modeling it is a one-time effort and it requires only infrequent changes, e.g. when new types of components, like solid-state drives in recent years, emerge . In its TBox, the ontology defines the available types of components and possible relations between those. It also defines the association between types and icons

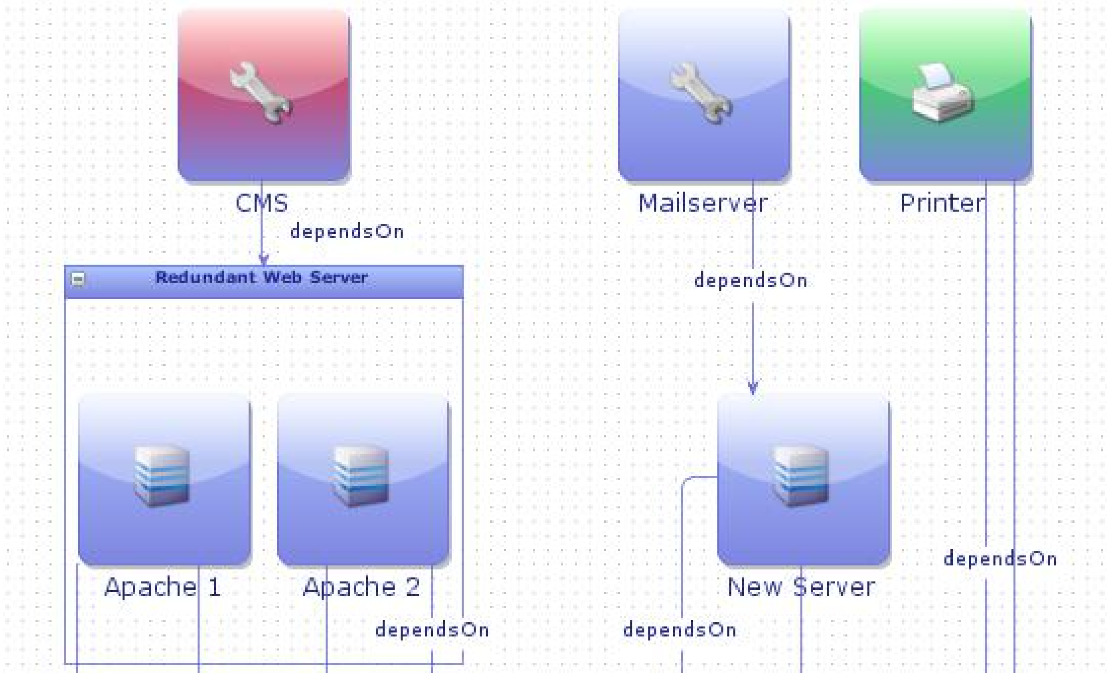


Figure 5.4.: Screenshot of the visual representation of the dependency network and provided evidence (green and red) about the availability of components.

shown in the user interface of RoCA. Whenever a user adds a new component, he has to select one of the types defined in the ontology. Thus, RoCA is highly customizable and not tied to a predetermined vocabulary. This eases the integration with modeling styles that are already in use within the organization, e.g., the terminology of the configuration management databases. As described earlier, relations can be constraint to be only allowed between certain types of components, to enforce a consistent model. A very popular tool for designing and modeling ontologies is Protégé (Musen, 2015). It provides customizable user interface to design and model ontologies. It also offers connections to reasoner for checking the consistency of the built ontology.

The dependency graph can be stored in the ABox of the ontology. If no dependency graph is specified, the user is presented with an empty model and can create a new graph there. Extending a graph loaded from the ABox is also possible.

### 5.6.2. User Interaction

User interaction with RoCA occurs in two different scenarios: when modeling the infrastructure as dependency graph, and when conducting a root cause analysis after an incident occurred.

When modeling the user can choose from components and relations specified in the background knowledge. Components and relations between them can be arrange by drag&drop or an automatic layout algorithm. The user can freely assign names for each component and the a priori weight in a details dialog. A first-order logic reasoner, e.g. Pellet (Sirin et al., 2007) or HermiT (Glimm et al., 2014) can check the finished model for consistency. The model can be saved and also exported as graph.

When an incident occurs, the user can load the previously created model and enter observations about available (marked as green) and unavailable (red) components. There can be entered any number of observations. In our example screenshot (Fig. 5.4) we have marked the components *CMS* as active (green) and *Printer* as inactive (red), while we have not specified any information related to the component *Mailserver*. If there is no information about some component, it is simply left as unknown. Once these observations have been specified, the user can run the root cause analysis directly from the interface and the most probable root cause is presented. By inspecting the most probable root cause and all assumptions that are entailed, the user might agree on the proposed root cause or might specify additional observations as shown in the workflow shown of Figure 5.2.

## 5.7. Estimating Availabilities in IT Infrastructures

Our approach has two major features. First, it employs a reusable top-level model of the infrastructure dependencies. Second, the measured availabilities are added to each infrastructure component. The use of measured availabilities frees us from the need to model each measured threat manually. If a new threat surfaces, it can be added to the model and expected changes to the availabilities can be calculated. In this chapter, we focus on impacts that make infrastructure components unavailable.

We start our approach by creating a dependency graph of the central components and services. To predict availabilities of IT components and services we need a way to represent (1) the logic of dependency of IT infrastructure, (2) probabilistic values for availabilities and (3) new threats. The dependencies in IT infrastructure can easily be modeled with first-order logic, but, first-order logic alone has no way to calculate how a threat influences the probability that an infrastructure component is available.

Markov logic networks (MLN)(Richardson and Domingos, 2006) offer a single representation of probability and first-order logic by adding weights to formulas. Together with a set of constants, the MLN specifies a probability distribution over possible worlds. Marginal inference computes the probability for specific values of variables in these possible worlds.

Thereby, we can create a reusable IT infrastructure model that includes infrastructure dependencies and measured availability. The model can be used to calculate a prediction how new threats affect the availability of IT components in an IT infrastructure. This provides a fast and quantitative solution to determine if a new threat is acceptable or needs to be mitigated.

We demonstrate and evaluate our solution in a case study where we implement the MLN for a small part of an IT infrastructure. We show how the change of availabilities through a new threat can be predicted and how to analyze a risk mitigation approach for this threat.

### 5.7.1. Availability

Achieving high availability, defined as up to 5 minutes unavailability per year, is a long-standing goal in the IT industry (Gray and Siewiorek, 1991). Continuous monitoring of availability is part of IT service management best practices like the Information Technology Infrastructure Library (ITIL) (Cabinet Office, 2011). We define availability as the probability that a system is working properly and reachable. The availability of a system can be determined as follows:

$$Availability = \frac{Uptime}{Uptime + Downtime} \tag{5.9}$$

Unavailability is the inverse of availability:

$$Unavailability = 1 - Availability \tag{5.10}$$

We distinguish between measured availability and predicted availability. The measured availability of important infrastructure components and services is typically measured over a one-year time-frame. The predicted availability is an estimation how the availability will be, under the assumption of specific changes in the threat-landscape or the infrastructure. We use marginal inference in Markov logic networks to calculate the predicted availabilities in our approach.

### 5.7.2. Summary of the Approach

We start by creating a dependency network of all major infrastructure components and services. We transform this network into evidence for our MLN program. We collect unavailability information for each node and use a learning algorithm to add the corresponding weights to the evidence. By adding threats, the resulting evidence can be used to determine the effect of a local threat to the whole network.

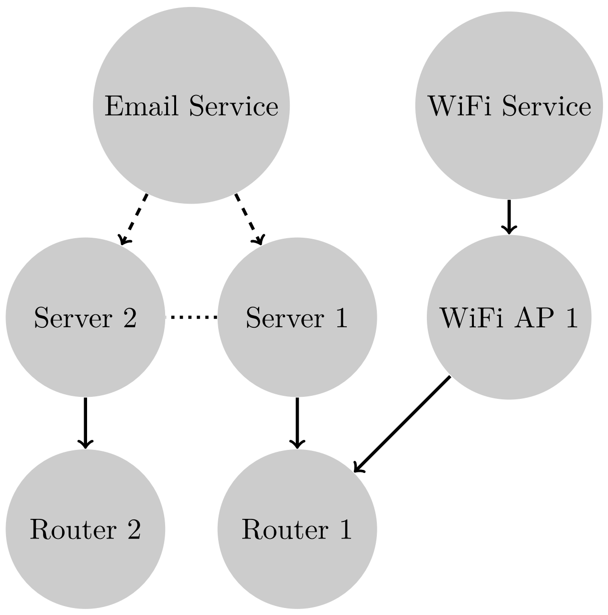


Figure 5.5.: The dependency network of the small IT infrastructure of our case study. Solid arrows indicate specific dependencies, dashed arrows symbolize generic dependencies and the dotted line represents a redundancy.

### 5.7.3. Case Study

A small case study demonstrates the usability of our solution. There exist several open-source inference engines for Markov logic networks, e.g. Alchemy[5], Tuffy[6], and RockIt[7]. We use RockIt for the calculations in our study.

The base configuration of the dependency network (see Figure 5.5) has seven nodes: An `Email Service` is realized by two redundant servers `Server 1` and `Server 2`. Each of the servers depends on its own router (`Router 1` and `Router 2`, respectively). One of the two routers is used by a `WiFi Access Point (AP)`, which offers a `WiFi Service`. The corresponding evidence is listed in Table 5.6. The infrastructure components and services have the measured unavailabilities given in column Scenario 1 in Table 5.2 and the corresponding, learned weights are shown in Table 5.3.

We can now add a new threat: `Router 1` threatens to overheat because construction work cut off the normal airflow to the room. The predicate expressing this is `endangered("Overheating", "Router 1", 3)`. By using marginal inference, we get the offline probabilities, which give us the predicted unavailabilities shown in column Scenario 2 in Table 5.2.

[5]http://alchemy.cs.washington.edu
[6]http://hazy.cs.wisc.edu/hazy/tuffy
[7]https://code.google.com/p/rockit

Figure 5.6.: The predicates for the dependency network of the base configuration.

```
genericDependency("Email Service","Server 1")
genericDependency("Email Service","Server 2")
redundancy("Server 1","Server 2")
specificDependency("Server 1","Router 1")
specificDependency("Server 2","Router 2")
specificDependency("WiFi AP 1","Router 1")
specificDependency("WiFi Service","WiFi AP 1")
```

Table 5.2.: The measured and predicted unavailabilities for the different scenarios of the case study. Scenario 1 is the base configuration, Scenario 2 is the base configuration with the overheating threat, Scenario 3 is the configuration with a second WiFi AP, and Scenario 4 is the configuration with a second WiFi AP and the overheating threat.

| Component | Scenario 1 (measured) | Scenario 2 (predicted) | Scenario 3 (measured) | Scenario 4 (predicted) |
|---|---|---|---|---|
| `Email Service` | 0.0010 | 0.0010 | 0.0010 | 0.0010 |
| `Router 1` | 0.0005 | 0.0100 | 0.0005 | 0.0099 |
| `Router 2` | 0.0010 | 0.0009 | 0.0010 | 0.0009 |
| `Server 1` | 0.0015 | 0.0110 | 0.0015 | 0.0109 |
| `Server 2` | 0.0020 | 0.0019 | 0.0020 | 0.0019 |
| `WiFi Service` | 0.0015 | 0.0110 | 0.0015 | 0.0015 |
| `WiFi AP 1` | 0.0010 | 0.0105 | 0.0010 | 0.0104 |
| `WiFi AP 2` | - | - | 0.0010 | 0.0010 |

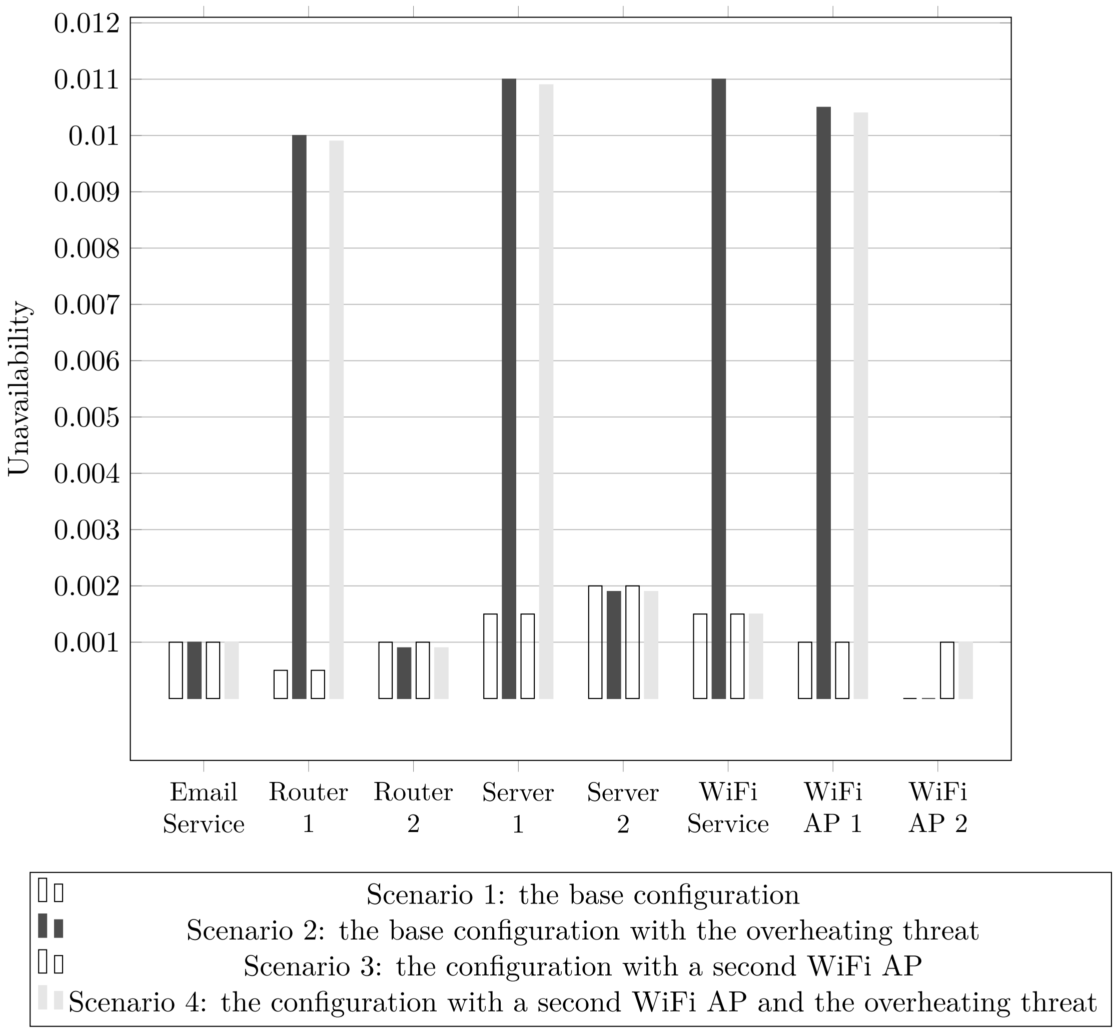
The threat increased the unavailability of `Router 1` by 0.0095. This has the effect that the unavailability of `WiFi AP 1`, `Server 1` and `WiFi Service` also increases by 0.0095. On the other hand, the unavailability of `Email Service` remains unaffected, because it can use `Server 2`, which depends on `Router 2`. The changes can be seen in Figure 5.7.

The cooling problem can only be solved through expensive additional construction work; therefore, other risk mitigation approaches should be investigated.

A cheap risk mitigation approach is to provide a second, redundant `WiFi AP 2`, which uses `Router 2`. We update the infrastructure model (Table 5.4) and set the unavailability for the new `WiFi AP 2` equal to that of `WiFi AP 1` (see Scenario 3 in Table 5.2). Again we use the MLN program as described above and learn the corresponding weights (Table 5.3).

Now we can add again the threat `endangered("Overheating", "Router 1", 3)` and calculate the unavailabilities (see Scenario 4 in Table 5.2) with the help of marginal inference. The unavailability of `Router 1` increases by 0.0094. The unavailability of

Figure 5.7.: The change of the unavailabilities in the different scenarios of our case study.



`Server 1` and `WiFi AP 1` also increases again but this time the unavailability of the `WiFi Service` remains unaffected. The changes are also shown in Figure 5.7.

We can see that a second WiFi Access Point would successfully mitigate the risk for the WiFi Service.

The slightly different increase of the unavailability in the two scenarios has two reasons. First, the learning algorithm provides only an approximation of the correct weights for the unavailabilities. Second, the number of evidence (and thereby the number of possible worlds) changed between the two scenarios, but the weight of the threat remained the same. As can be seen in Figure 5.7, these effects are small enough to not influence the overall result.

Table 5.3.: The learned weights for the measured unavailabilities in the base configuration.

```
measuredUnavailability("Email Service",-6.907250)
measuredUnavailability("Router 1",6.933551)
measuredUnavailability("Router 2",0.001400)
measuredUnavailability("Server 1",-6.908549)
measuredUnavailability("Server 2",-6.905200)
measuredUnavailability("WiFi AP 1",-0.023739)
measuredUnavailability("WiFi Service",-7.590026)
```

Table 5.4.: The predicates for the dependency network with a second WiFi AP.

```
genericDependency("Email Service","Server 1")
genericDependency("Email Service","Server 2")
genericDependency("WiFi Service","WiFi AP 1")
genericDependency("WiFi Service","WiFi AP 2")
redundancy("Server 1","Server 2")
redundancy("WiFi AP 1","WiFi AP 2")
specificDependency("Server 1","Router 1")
specificDependency("Server 2","Router 2")
specificDependency("WiFi AP 1","Router 1")
specificDependency("WiFi AP 2","Router 2")
```

The case study also demonstrates that the results of MLN marginal inference are not exactly the results one would expect from regular probability calculation. However, as we demonstrate, the MLN calculation is well suited for the calculation of threat propagation.

## 5.8. Related Work

Related work can roughly be divided into two parts: Approaches also conducting root cause analysis, but using a different method; and approaches using probabilistic frameworks for abductive reasoning, yet not in the context of root cause analysis.

### 5.8.1. Root Cause Analysis

In previous work, failure diagnosis is conducted using correlation measures. A specific correlation measure for failure diagnosis is presented in (Marwede et al., 2009). The

Table 5.5.: The new learned weights for the configuration with a second WiFi AP.

```
measuredUnavailability("Email Service",-6.906300)
measuredUnavailability("Router 1",6.909860)
measuredUnavailability("Router 2",10.333070)
measuredUnavailability("Server 1",-6.905100)
measuredUnavailability("Server 2",-6.888114)
measuredUnavailability("WiFi AP 1",-7.600700)
measuredUnavailability("WiFi AP 2",-10.364995)
measuredUnavailability("WiFi Service",-6.500500)
```

approach uses anomalies in the timing of program calls to trace the real root cause of an event. The anomalies are aggregated to give an anomaly score for each component. The scores are correlated within their architectural level to determine an anomaly ranking, which expresses the likelihood that a component is the root cause of a failure. A method for failure diagnosis using decision trees is proposed in (Chen et al., 2004). The decision tree classifies the successful as well as failed requests. A correlation of paths in the decision tree with occurred failures indicates the node that represents the likely root cause.

In (Zawawy et al., 2012) an approach for requirements-driven root cause analysis for failures in software systems is proposed, wherein a Markov Logic Network is used as knowledge repository for diagnostic knowledge. The approach uses log data as observation information, the Markov Logic Network is used to deal with uncertainty stemming from incomplete log data. Their approach differs from ours in several points: they first model the background knowledge as goal trees and only convert it to first-order logic later; the evidence is solely generated from log data; and most importantly they use marginal inference, different to our approach which uses MAP inference. In (von Stülpnagel et al., 2014) marginal inference was also used for the purpose of estimating unavailabilities in an IT infrastructure, where the authors referred to problems when marginal inference is applied to very low probabilities usually attached to the occurrence of risks in an IT setting. These problems are based on the use of sampling algorithms for performing marginal inference, as exact inference is infeasible Our approach is based on solving an optimization problem, which is not affected negatively by very small probabilities.

The Shrink tool (Kandula et al., 2005) uses a Bayesian Network to model the diagnosis problem. It extends previous work on fault diagnosis with Bayesian Networks (Steinder and Sethi, 2002), by proposing a greedy inference algorithm with polynomial running time. Furthermore, Shrink is able to handle noise and small inaccuracies in the observations.

We made efforts to apply the approaches referred to in the previous section to model and analyze the scenarios mentioned in Section 5.4. However, we had to retract from this task for several reasons:

- In most of the cases the cited literature described an approach with a focus on its theoretical foundations without offering a tool that implements the approach.
- In other cases it is crucial to understand that we would have to compare apples and oranges because the proposed techniques cannot be applied to solve the task that we try to solve with our tool.

In the following we will describe these issues in detail.

As mentioned above, Heiden et al. (2017) propose an approach that is very close to ours in the sense that it is also based on the idea of abductive reasoning. As we already mentioned, the approach does not support probabilities for threats nor does it support probabilities for any another aspect. This means that in each situation where there are several possible root causes, the approach proposed by Heiden et al. will not find the most probable cause, but only one of the possible causes (or all of them). The probabilistic knowledge used in our approach cannot be expressed within the approach of Heiden et al. While the approach also supports the dependencies we expressed in Formulas 5.2a to 5.2f, risk probabilities that we express in formulas as 5.3b and 5.3c, which can be derived from the statistical data gathered by monitoring systems (see the end of Section 5.3.2) or from background knowledge as described in Section 5.3.3), cannot be expressed and will thus not influence the computation of the root cause. However, such information is obviously crucial and will help to distinguish between several possible reasons for a failure. While the approach of Liu and Chiou (1997) is based on a completely different formalism, it suffers from the same problems, namely its incapability to model probabilistic knowledge.

This is different when we look at the systems that are based on Bayesian Networks and their extensions. This comprises the works of Weidl et al. (2005) as well as the approach implemented in the Shrink tool Kandula et al. (2005). The main differences between our approach and theirs are based in the models used which correspond to a relational representation for our approach, in contrast to a propositional representation in Weidl's et al. In a relational representation, we can easily write down formulas like 5.2a to 5.2f, or 5.6.These formulas are general formulas that use variable. This makes it rather convenient to express general dependencies. Even though the general formulas are finally grounded in order to compute a most probable root cause, on the representation level we describe an infrastructure and its dependencies with the help of relations and general formulas. This is much more complicated in an approach that is based on a propositional representation. Here we cannot write down a simple formula as Formula 5.2. Instead we have to add an explicit edge in the Bayesian Network, for each possible instantiation of the variables in the formula.

Another major difference is based in the directedness of Bayesian Networks. When trying to model an IT infrastructure and its potential root causes, the modeling approach in a directed graphical model will always be guided by potential errors that cause a problem somewhere in the infrastructure. That means that the development of the network will be guided by the attempt to model causal relationships. This is not the case in an undirected model that is based on the observations of correlations without making assumptions about causal dependencies. In such a model it is much easier to integrate probabilities that are gathered by statistical observations.

### 5.8.2. Applications of Abductive Reasoning

In (Singla and Mooney, 2011), Singla et al. extend the approach presented in (Kate and Mooney, 2009) and use it in the context of plan and intent recognition. Instead of adding reverse implication, they introduce a hidden cause for all implications with the same left-hand side. In general, this reduces the size of the MLN and subsequently increases performance. However, as detailed above, for our approach the mutual exclusivity clauses are not needed anyway. Nonetheless, if more probable events have to be included in the evidence, their optimization can also be included in our approach.

Most other approaches to abductive reasoning either use first-order logic to calculate a minimal set of assumptions sufficient to explain the hypothesis (Kakas et al., 1992; Ng and Mooney, 1991; Poole et al., 1987; Stickel, 1991), or Bayesian Networks to compute the posterior probability of alternative explanations given the observations (Pearl, 1988). The former approaches are not able to estimate the likelihood of alternative explanations, as they do not support uncertainty in the background knowledge or evidence. Bayesian Networks, on the other hand, are designed to handle uncertainty. However, as they are propositional in nature, they cannot handle structured knowledge involving relations amongst multiple entities directly (Kate and Mooney, 2009).

Bayesian Abductive Logic Programs (BALP) (Raghavan and Mooney, 2010) are another approach that combines first-order logic and probabilistic graphical models. The main difference to MLNs is that BALPs are based on Bayesian Networks, which are directed. Undirected relations, like the symmetry of redundancy, are thus more complex to model. Inoue and Inui (2011) describe a system that uses integer linear programming (ILP) for weighted abduction. They outperform a state-of-the-art abductive engine (Mini-TACITUS (Ovchinnikova et al., 2014)). The MLN solver we use also transforms the problem internally to an ILP, which is one of the reasons for its good runtime performance.

### 5.8.3. Estimating Availability

There exist alternative approaches that combine logic and probability (see (Braz et al., 2008) for an overview). Here we limit the discussion of related work to Bayesian logic programs (BLP) (Kersting and De Raedt, 2001). Because of the usage of Bayesian networks in combination with risk analysis (Weber et al., 2012), BLPs seem to be the most prevalent alternative to MLNs.

Bayesian logic programs aim at resolving some limitations of Bayesian networks, among others the essentially propositional nature of their representations. Therefore, Bayesian logic programs unify Bayesian networks with logic programming. Each Bayesian logic program represents a Bayesian network by mapping the atoms in the least Herbrand model, which constitutes the semantics of the logic program, to nodes of the Bayesian network.

A BLP computes the probability for one query from the known probabilities in the resolution to the query statement. In contrast, MLNs infer all probabilities from a given complete set of weights.

Furthermore, there exist efficient solvers for MLNs and they have been applied to a wide range of problems, for instance in requirements-driven root cause analysis (Zawawy et al., 2012) and data integration (Niepert et al., 2011a).

Another advantages of MLNs over other probabilistic logic approaches is that the Markov networks, constructed through the MLN, are undirected graphs. This has the effect that changing the weights at one node will influence the whole graph and thereby the results. Hence, it is more likely to find new relationships between elements that where not explicitly modeled. On the other side, this has the disadvantage that it is harder to isolate a single variable. Unlike BLPs (Kersting and De Raedt, 2001), MLNs have very simple semantics while theoretically keeping the expressive power of first-order logic (Richardson and Domingos, 2006).

While risk management in general makes increasing use of Bayesian networks (Weber et al., 2012), we are not aware of any automatic or semi-automatic approach for IT risk management. There exist numerous IT risk management frameworks and standards. The European Union Agency for Network and Information Security (ENISA) lists 17 different ones in their Inventory of Risk Management[8]. One example on a national level is the German *IT-Grundschutz Methodology* (Bundesamt für Sicherheit in der Informationstechnik, 2008). It is a qualitative method and assumes that threats to the secure operation of information processing are similar for all types of organizations. It provides the *IT-Grundschutz Catalogue* (Bundesamt für Sicherheit in der Informationstechnik, 2016), which describes threats for typical areas of application and IT components, based on generic risk analysis performed by the German Federal Office for Information Security (Bundesamt für Sicherheit in der Informationstechnik). The *IT-Grundschutz Catalogue* also provides standard security measures for the typical objects that are affected by these threats. A detailed risk analysis, where the probability of a threat is determined, is not part of the *IT-Grundschutz Methodology* but it is described how it can be incorporated.

There are other approaches that use dependency networks to determine the global impact of a threat. The approach of Zambon et al. (2009) focuses on downtime and temporal aspects but unlike our approach it is qualitative and considers only one threat at a time. The approach of Breu et al. (2008) uses the number of attacks as the key quantitative

[8]`http://rm-inv.enisa.europa.eu/methods/rm_ra_methods.html`

concept and models each threat separately but does not take redundant components into account.

## 5.9. Discussion and Conclusion

We presented our approach of applying abductive reasoning using Markov Logic Networks to compute the most probable root cause for a failure in an IT infrastructure. Our approach models the infrastructure with the help of ontologies. In particular, we formulated the dependencies of the network as hard formulas. Moreover, we added weighted soft formulas to model the probability of risks that might result in the failure of components and services. We defined these risks in accordance to the taxonomy of the *IT-Grundschutz Catalogues*. Furthermore, we argued how the expressiveness of ontologies can be used to model general, reusable knowledge concerning risks and IT components. Our approach uses the same formalism for both knowledge presentation and abductive reasoning. Thus, all relevant information is readily available to compute the most probable root cause once an incident occurs. To the best of our knowledge, there exists no other approach that combines uncertainty and logical abductive reasoning to solve the problem of root cause analysis. We implemented our approach in RoCA, a tool providing a graphical user interface for modeling the infrastructure and conducting the root cause analysis.

We conducted an evaluation of our approach by analyzing two failures that happened in the infrastructure of our research group. In both cases we were able to determine a root cause (respectively, a sequence of probable root causes) that turned out to be helpful for a system administrator to resolve the problem. Our approach is especially useful when the reasons for the failure are not obvious to the administrator that is in charge of resolving the problem. Thus, our approach will be more beneficial in IT infrastructures, where competences are scattered over the members of different organizational units.

Furthermore, we analyzed the scalability of the approach for various sizes of randomly generated infrastructures, modeled after properties observed in real-world scenarios. The approach proofed to scale linearly in the size of the infrastructure up to hundreds of thousands of individual components.

Finally, we want to stress that the presented approach is not only suited for technical hardware scenarios, but can almost effortlessly be transferred to other settings. One example would be the search for an index case (patient zero) during the outbreak of a disease Mohammadi (2015). Relation between person are modeled similarly, however those will also have weights, representing the uncertainty that people actually know each other or where in contact during a time period. By noting when somebody showed the first symptoms and verifying that the person with the earliest symptoms can have had contact (directly or indirectly through others) with the other patients, the index case can be identified.

To our knowledge this work presents the first application of Markov logic networks to risk management. We have shown that MLNs generally allow an automatic calculation of risks, exemplified by the probability of availability, in an IT infrastructure.

As described in the case study, our solution allows us to efficiently calculate how a threat affects an infrastructure network. We have demonstrated how the analysis can be used to compare changes in the infrastructure and thereby support the risk mitigation. Our approach has two major features. First, it employs a top-level model for the dependency network, which can be easily maintained and reused. Second, it uses the measured availabilities of the infrastructure components and thereby creates a quantitative infrastructure model without modeling every threat separately. This also has the advantage that the manual work for risk analysis is greatly reduced.

In most cases, risk assessments are performed regularly, e.g. every six month. At the same time, the link between availability and risk management is seldom used and technological support for IT risk management was identified as one area where improvement is needed (Ernst & Young, 2013).

Our approach enhances risk assessment by providing an automation and allowing reuse of earlier work. By using the availability measurements of the IT service management as quantitative input and provide results in the same way, we allow an easy communication of IT risk.

However, there are some limitations, caveats and lessons learned from using Markov logic networks, and in particular marginal inference in MLNs.

It is not possible to simply add or remove infrastructure components or services to or from an existing model. This is so because the weight of the new component or service influences the total weight of the model, which in turn affects the probabilities of all statements. In the current state of our solution, adding or removing a new infrastructure component requires relearning all weights.

Even though MLNs use a quite simple representation, modeling with them is not as straightforward as it may seem (Jain, 2011). Each variable – every literal that involves a hidden predicate – must have a weight. For instance in our case study this means that each node in the dependency network needs a `measuredUnavailability(infra, float)`. This requires an accurate specification of the MLN evidence. However, since the measurement of the availability is a best practice (Cabinet Office, 2011), the necessary data should be available.

Without an understanding of the normalization of weights and the log-linear model, the relationship between the weights and the resulting probabilities of the marginal inference can be counterintuitive. Changing the weight of a single formula shifts the relative weights of the possible worlds according to where the formula is true. This results in sometimes unexpected changes in probabilities of statements. Therefore, we usually learn the weights for the measured unavailabilities. While modeling with MLNs it is also important to have in mind that the weight of a soft formula does not directly correspond to a specific universal probability for this formula. The probability depends on the whole MLN and the modeled domain, including the number of individuals in this domain.

A limitation of existing implementations for marginal inference is their use of sampling algorithms to calculate the probabilities. To determine the effect of threats with very low probabilities, which are quite common in risk management, sampling requires a high number of iterations. Threats with very small probabilities can provide the problem that their frequency is so low that they do not influence the availability measurement of a component. These threats can still be added manually through the predicate `endangered(threat,infra, float)`. To our knowledge, there is no alternative to sampling as the exact computation is too complex to be done for larger networks. However, it has the advantage that it allows us to choose how much time to invest into the accuracy of the probabilities. We reduce the problem of sampling with low probability by using the measured unavailability, as combination of many small threats and thereby are able to aggregate them to larger numbers.

While there are different MLN solvers, to our knowledge none of them supports full first-order logic. Because of the undecidability of first-order logic, this will most likely not change.

So far, we have not taken all risks of the Grundschutz Catalogues into account. Instead, we have focused on a subset relevant for the infrastructure we modeled. To apply our approach to an arbitrary IT infrastructure, we have to create a complete translation of the catalogues to our logical representation.

# Part III.

# Conclusion and Future Work

# 6 Conclusion

As stated in the introduction, there is a lot of work on crisp logical knowledge representation – e.g. description logic and ontologies – and statistical representation of uncertainty. In recent years, there were efforts to bridge the gap between those two areas and harness the vast information contained in uncertain knowledge.

In this dissertation we investigated how uncertainty in large-scale knowledge graphs can be handled efficiently, and how bridging the gap between uncertainty in knowledge graphs and logical reasoning can improve the overall usefulness of the information. In the following sections we briefly recapitulate the results of each chapter and draw conclusions, especially regarding the objectives set in Section 1.2:

1. debugging uncertain (temporal) knowledge bases in order to generate consistent knowledge graphs to make them accessible for logical reasoning,
2. combining probabilistic query answering and logical reasoning which in turn uses those consistent knowledge graphs to answer user queries, and
3. employing the aforementioned techniques to the problem of risk management in IT infrastructures, as a concrete real-world application.

## 6.1. Extracting Consistent Knowledge Graphs from Uncertain Information

In Chapter 3 we used probabilistic graphical models – namely Markov logic networks (MLN) and probabilistic soft logic (PSL), together with a numerical extension – to define rules and constraints over uncertain (temporal) knowledge graphs. Our approach uses probabilistic graphical models for reasoning and inference over those uncertain knowledge graphs. We have shown that this can be used to

- infer new knowledge from existing facts,
- detect erroneous facts,
- compute the most probable consistent knowledge graph, regarding those rules.

All those tasks naturally account for the uncertainty inherent to the given facts. Although these problems are NP-hard and #P-hard, respectively, our experiments show that our approach produces results in an acceptable amount of time, even on large-scale knowledge graphs such as Wikidata. The experiments also demonstrated that the usage of probabilistic graphical models to do reasoning over uncertain (temporal) knowledge graphs produces viable results, even in very noisy settings.

## 6.2. Assessing the Real-World Usability of the Combination of Probabilistic and Logical Reasoning through Query Rewriting

The work presented in Chapter 4 provides a thorough view on handling large-scale, uncertain data with Semantic Web technologies. We first analyzed queries from the LSQ dataset and found that from those queries that can directly be translated to union of conjunctive queries, almost all are safe. From this we conclude that the information needs users have – expressed through their queries – are in general feasible to be fulfilled in tractable time over probabilistic data.

Next, we described our preliminary implementation towards a probabilistic OBDA system for large-scale knowledge bases. It combines tractability for the class of safe queries with the benefits of ontology-based query rewriting. To demonstrate the good scalability of the approach, we compared its performance to ProbLog, a state-of-the-art inference system. As benchmark datasets we used NELL, representing a real-world dataset, and additionally created a probabilistic extension of the LUBM benchmark that allows us to create datasets of any size to measure scalability. We have shown that it scales well compared to the other system and is able to compute query answers in a reasonable amount of time for knowledge bases containing several million facts.

To the best of our knowledge this is the first work that analyzes real-world SPARQL queries in the light of probabilistic data, and evaluates the real-world performance of query rewriting in probabilistic databases.

## 6.3. IT Risk Management in Large-scale IT Infrastructures using Probabilistic Models

In Chapter 5 we applied Markov logic networks in the setting of IT infrastructure and risk management. Our approach uses abductive reasoning for the MAP state to compute the most probable root cause for a failure in an IT infrastructure. We modeled the background knowledge and formulated dependencies between components and threats affecting those as logical formulas. This allows for a general reusable model of the IT infrastructure, which can easily adjusted to other infrastructures and bootstrap the modeling there. Reasoning in MLNs is by default deductive. However, we developed specialized rules which allow us to conduct abductive reasoning in the setting of risk management. As our approach uses first-order logic for both modeling and reasoning, all relevant information is readily available to compute the most probable root cause once an incident occurs.

To evaluate the usefulness of the approach, we analyzed two failure scenarios that occurred in the infrastructure of our research group and the computing center of the university. Both cases showed that the determined root causes (or sequences thereof for diagnosis) were helpful for a system administrator to quickly determine the source of the failure and resolve the problem. In cases where the reasons for the failure are not obvious or the administrator is not familiar with the exact part of the infrastructure where the outage arises, our approach proved to be especially valuable. To evaluate the scalability of the approach, we implemented a generator to automatically create infrastructures of various sizes, which are modeled after real-world observations. The approach proofs to scale linearly in the size of the infrastructure up to hundreds of thousands of individual components. Additionally, we did some experiments using marginal inference to estimate the availability of components (measured in % uptime).

We also implemented a ready-to-use tool for our approach, providing a graphical user interface for modeling and conducting the root cause analysis. To the best of our knowledge, there exists no other approach that combines uncertainty and logical abductive reasoning to solve the problem of root cause analysis.

Finally, we want to stress that the presented approach is not only suited for technical hardware scenarios, but can almost effortlessly be transferred to other settings. One example would be the search for an index case (also known as *patient zero*) during the outbreak of a disease Mohammadi (2015). Relations between persons are modeled similarly, however those will also have weights, representing the uncertainty that people actually know each other or were in contact during a time period. By noting the time when somebody showed the first symptoms and verifying that the person with the earliest symptoms potentially had contact (directly or indirectly through others) with the other patients, the index case can be identified. To our knowledge this work presents the first application of Markov logic networks to risk management. We have shown that MLNs generally allow an automatic calculation of risks, exemplified by the probability of availability, in an IT infrastructure.

## 6.4. Closing Remarks

Overall, we have shown that uncertainty can be handled efficiently in multiple scenarios where current approaches do ignore or do not fully incorporated it.

The main result, which we could show in all three settings is the following: Despite the theoretical intractability of most of the used formalisms – reasoning in Markov logic networks is NP-hard; inference in probabilistic database is #P-hard except for the special case of safe queries – in real-world settings, like debugging existing uncertain knowledge graphs, answering user queries, or analyzing threats to an infrastructure, those approaches scale well, as the full expressiveness of those technologies is rarely used.

The results show that continuing research in that direction is both worthwhile and feasible in practice: Embracing uncertainty increases the usefulness of results and provides users with more worthwhile information.

# 7 Future Work

Each of the three investigated areas leaves options for further research. In this chapter we describe potential directions and benefits of future work.

## Debugging Large-scale Uncertain Temporal Knowledge Graphs

In Chapter 3 we only considered *valid time*. A straightforward extension is to either look only at *transaction time*, or extend the approach to be *bi-temporal*, i.e., include valid time and transaction time. For example, this allows for rules between valid and transaction time, like the valid time of a birthdate has to be before its transaction time (in simpler words, one can not specify a concrete birthdate in the future).

Similarly, instead of having two temporal dimensions, the approach can also be applied to spatial dimensions, opening up the ability to reason about locations, e.g. a capital has to be within the border of a country. Ultimately, all dimension can be combined to support and conduct spatio-temporal reasoning. Taking into account preferences Fionda and Pirrò (2013) is another line of future work.

In Section 3.4 we already introduced preliminary experiments with rule mining to automatically create consistency constraints. One line of work is to extend those experiments. Interesting questions are how those automatically mined rules work overall, or how easily those rules can be transfered to different or integrated knowledge graphs.

We also did not test the approach in a data integration scenario where multiple knowledge graphs are merged. Data integration is an active research area, where the expressive rules can help in both, finding inconsistencies between knowledge graphs, and providing one coherent view on the data (e.g. translating from one relation to another, possibly by combining statements, like for *first name* and *last name* being equal to *name*).

So far, we also only investigated possible applications of MAP inference. By using marginal inference, we can retain uncertain information in the computed results and ask for the probability of specific statements being valid.

Besides applying and transferring the approach to other scenarios, further improving scalability is also an objective. There is preliminary work in parallelizing MLN solvers and using highly optimized relational database systems to improve the overall performance.

An aspect we also did not cover here are the types of errors that were found in the actual knowledge graphs.

## Scalable Probabilistic Query Answering and Logical Reasoning

For future work on the research presented in Chapter 4, we propose several different directions. First, to further study queries using constructs like DISTINCT/GROUP BY, OPTIONAL, or FILTER and how they influence query safeness. Second, considering unsafe queries, to analyze their structure and investigate ways of handling those effectively, e.g. through approximation or simplification similarly to approaches presented for probabilistic databases. Furthermore, an interesting direction is to also address the problem of uncertain TBox elements.

To improve the evaluation of performance, a more thorough benchmark based on recent proposals for benchmarks specifically aimed at OBDA (e.g. (Lanti et al., 2015)) should be considered.

Finally, the tool to determine query safeness should be integrated in a probabilistic OBDA application that supports the complete workflow from mapping an ontology to one or more databases, over processing SPARQL queries, to a probabilistic result set. This would greatly facilitate the use of probabilistic query answering by non-expert users and in more real-world scenarios.

## IT Risk Management in Large-scale IT Infrastructures

From the modeling perspective, the work in Chapter 5 can be extended by including a more detailed description of risks and threats, as we up to now only considered a very simple model for those, and also only a small subset of the ones, e.g. listed in the *IT-Grundschutz Catalogues.*

Furthermore, as already outlined in previous chapter, the methodology for root cause analysis can also be employed and tested in different settings, like epidemiology to determine the index case of an outbreak.

# Bibliography

Abiteboul, Serge and Vianu, Victor. Datalog extensions for database queries and updates. *Journal of Computer and System Sciences*, 43(1):62–124, 1991. ISSN 10902724. doi: 10.1016/0022-0000(91)90032-Z.

Allen, James F. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.

Artale, Alessandro, Calvanese, Diego, Kontchakov, Roman, and Zakharyaschev, Michael. The DL-Lite Family and Relations. *Journal of Artificial Intelligence Research*, 36:1–69, 2009. ISSN 10769757. doi: 10.1613/jair.2820.

Baader, Franz and Nutt, Werner. Basic Description Logics. In *The Description Logic Handbook: Theory, Implementation, and Applications*, pages 43–95. Cambridge University Press, 2003.

Baader, Franz, Calvanese, Diego, McGuinness, Deborah L., Nardi, Daniele, and Patel-Schneider, Peter F. *The Description Logic Handbook: Theory, Implementation, and Applications*, volume 32. Cambridge University Press, 2003. ISBN 0521781760. doi: 10.2277/0521781760. URL `http://portal.acm.org/citation.cfm?id=1215128`.

Baader, Franz, Brandt, Sebastian, and Lutz, Carsten. Pushing the EL Envelope. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI)*, volume 5, pages 364–369, 2005. doi: 10.1097/01.NUMA.0000440635.64013.58. URL `http://www.ncbi.nlm.nih.gov/pubmed/24426756http://ijcai.org/papers/0372.pdfhttp://lat.inf.tu-dresden.de/research/reports.html`.

Bach, Stephen H, Broecheler, Matthias, Huang, Bert, and Getoor, Lise. Hinge-Loss Markov Random Fields and Probabilistic Soft Logic. *Journal of Machine Learning Research*, 18(109):1–67, 2017. URL `http://jmlr.org/papers/v18/15-631.html`.

Bail, Samantha, Alkiviadous, Sandra, Parsia, Bijan, Workman, David, Van Harmelen, Mark, Goncalves, Rafael S., and Garilao, Cristina. FishMark: A Linked Data Application Benchmark. *CEUR Workshop Proceedings*, 943:1–15, 2012. ISSN 16130073.

Banko, Michele, Cafarella, Michael J., Soderland, Stephen, Broadhead, Matt, and Etzioni, Oren. Open Information Extraction from the Web. *Proceedings of IJCAI-07, the International Joint Conference on Artificial Intelligence*, pages 2670–2676, 2007. ISSN 00010782. doi: 10.1145/1409360.1409378.

Berendt, B, Hollink, L, Hollink, V, Luczak-Rösch, M, Möller, K, and Vallet, D. USEWOD2011 - 1st International Workshop on Usage Analysis and the Web of Data. *Proceedings of the 20th International Conference Companion on World Wide Web (WWW)*, pages 305–306, 2011. doi: 10.1145/

1963192.1963324. URL http://www.scopus.com/inward/record.url?eid=2-s2.0-79955127036&partnerID=40&md5=6ec8c19cff2e223d891c92e7c6e236e7.

Bizer, Christian and Schultz, Andreas. The Berlin SPARQL Benchmark. *International Journal on Semantic Web and Information Systems*, 5(2):1–24, 2001. ISSN 1552-6283. doi: 10.4018/jswis.2009040101.

Bizer, Christian, Heath, T, and Berners-Lee, T. Linked data-the story so far. *International Journal on Semantic Web and Information Systems*, 5(3):1–22, 2009. ISSN 15526283. doi: 10.4018/jswis.2009081901. URL http://eprints.soton.ac.uk/271285/.

Bollacker, Kurt, Evans, Colin, Paritosh, Praveen, Sturge, Tim, and Taylor, Jamie. Freebase: a collaboratively created graph database for structuring human knowledge. *SIGMOD 08 Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250, 2008. ISSN 07308078. doi: 10.1145/1376616.1376746. URL http://doi.acm.org/10.1145/1376616.1376746.

Braz, Rodrigo de Salvo, Amir, Eyal, and Roth, Dan. A Survey of First-Order Probabilistic Models. In Holmes, D E and Jain, L C, editors, *Innovations in Bayesian Networks*, volume 156 of *Studies in Computational Intelligence*, pages 289–317. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-85065-6. doi: 10.1007/978-3-540-85066-3_12. URL http://dx.doi.org/10.1007/978-3-540-85066-3_12.

Breu, Ruth, Innerhofer-Oberperfler, Frank, and Yautsiukhin, Artsiom. Quantitative assessment of enterprise security system. In *Availability, Reliability and Security, 2008. ARES 08. Third International Conference on*, pages 921–928. IEEE, 2008.

Bundesamt für Sicherheit in der Informationstechnik. IT-Grundschutz Methodology. Technical report, Bundesamt für Sicherheit in der Informationstechnik, 2008.

Bundesamt für Sicherheit in der Informationstechnik. Threat Catalogues. *IT Grundschutz Catalogues*, 15. EL:443 – 1338, 2016. URL https://www.bsi.bund.de/EN/Topics/ITGrundschutz/Download/download_node.html.

Cabinet Office. *ITIL Service Design.* TSO (The Stationery Office), 2011.

Cafarella, Michael J, Halevy, Alon, and Madhavan, Jayant. Structured data on the Web. *Communications of the ACM*, 54(2):72–79, 2011.

Calvanese, Diego, De Giacomo, Giuseppe, and Lembo, Domenico. DL-Lite: Tractable description logics for ontologies. *AAAI*, 5:602–607, 2005. URL http://www.aaai.org/Papers/AAAI/2005/AAAI05-094.pdf.

Calvanese, Diego, Giacomo, Giuseppe, Lembo, Domenico, Lenzerini, Maurizio, and Rosati, Riccardo. Tractable Reasoning and Efficient Query Answering in Description Logics: The DL-Lite Family. *Journal of Automated Reasoning*, 39(3):385–429, jul 2007. ISSN 0168-7433. doi: 10.1007/s10817-007-9078-x. URL http://link.springer.com/10.1007/s10817-007-9078-x.

Calvanese, Diego, Cogrel, Benjamin, Kalayci, Elem Guzel, Komla-Ebri, Sarah, Kontchakov, Roman, Lanti, Davide, Rezk, Martin, Rodriguez-Muro, Mariano, and Xiao, Guohui. OBDA with the Ontop Framework. *23rd Italian Symposium on Advanced Database Systems, SEBD 2015, Gaeta, Italy, June 14-17, 2015.*, pages 296–303, 2015.

Calvanese, Diego, Cogrel, Benjamin, Komla-Ebri, Sarah, Kontchakov, Roman, Lanti, Davide, Rezk, Martin, Rodriguez-Muro, Mariano, and Xiao, Guohui. Ontop : Answering SPARQL Queries over Relational Databases. *Semantic Web*, 8(3):471–487, 2017. ISSN 22104968. doi: 10.3233/SW-160217.

Carlson, Andrew, Betteridge, Justin, and Kisiel, Bryan. Toward an Architecture for Never-Ending Language Learning. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*, pages 1306–1313, 2010. ISBN 9781577354666. doi: 10.1002/ajp.20927. URL `http://www.aaai.org/ocs/index.php/AAAI/AAAI10/paper/download/1879/2201`.

Ceylan, İsmail İlkan and Peñaloza, Rafael. The Bayesian Ontology Language BEL. *Journal of Automated Reasoning*, 58(1):67–95, 2017. ISSN 15730670. doi: 10.1007/s10817-016-9386-0.

Chekol, Melisachew W., Pirrò, Giuseppe, Schoenfisch, Joerg, and Stuckenschmidt, Heiner. TeCoRe: Temporal Conflict Resolution in Knowledge Graphs. In *Proceedings of the VLDB Endowment*, volume 10, pages 1929–1932. VLDB Endowment, 2017a. doi: 10.14778/3137765.3137811. URL `http://dl.acm.org/citation.cfm?doid=3137765.3137811`.

Chekol, Melisachew Wudage, Huber, Jakob, Meilicke, Christian, and Stuckenschmidt, Heiner. Markov logic networks with numerical constraints. In *Frontiers in Artificial Intelligence and Applications*, volume 285, pages 1017–1025, 2016. ISBN 9781614996712. doi: 10.3233/978-1-61499-672-9-1017.

Chekol, M.W., Pirrò, G., Schoenfisch, J., and Stuckenschmidt, H. Marrying uncertainty and time in knowledge graphs. In *31st AAAI Conference on Artificial Intelligence, AAAI 2017*, 2017b.

Chen, M., Zheng, A.X. X, Lloyd, J., Jordan, M.I. I, and Brewer, E. Failure diagnosis using decision trees. In *International Conference on Autonomic Computing, 2004. Proceedings.*, pages 36–43. IEEE, 2004. ISBN 0-7695-2114-2. doi: 10.1109/ICAC.2004.1301345. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1301345`.

Chen, Willy, Hess, Claudia, Langermeier, Melanie, von Stuelpnagel, Janno, and Diefenthaler, Philipp. Semantic Enterprise Architecture Management. In *15th International Conference on Enterprise Information Systems (ICEIS)*, page 8, 2013.

Chen, Yang and Wang, Daisy Zhe. Knowledge expansion over probabilistic knowledge bases. *Proceedings of the 2014 ACM SIGMOD international conference on Management of data - SIGMOD '14*, pages 649–660, 2014. ISSN 07308078. doi: 10.1145/2588555.2610516. URL http://delivery.acm.org/10.1145/2620000/2610516/p649-chen.pdf?ip=132.234.102.220&id=2610516&acc=ACTIVESERVICE&key=65D80644F295BC0D.B53287412424DA9D.4D4702B0C3E38B35.4D4702B0C3E38B35&CFID=996856472&CFTOKEN=84796420&__acm__=1508474407_865aa23a3ad06e886b76b.

da Costa, PCG and Laskey, KB. PR-OWL: A framework for probabilistic ontologies. *Frontiers in Artificial Intelligence and ...*, 2006. URL http://books.google.com/books?hl=en&lr=&id=qqGx2ulX6hEC&oi=fnd&pg=PA237&dq=PR-OWL+:+A+Framework+for+Probabilistic+Ontologies&ots=Qz2DhfQMtu&sig=TU3Z5SY0iCG81rSJ2hmm2CFaWHQ.

Dalvi, Nilesh and Suciu, Dan. The Dichotomy of Probabilistic Inference for Unions of Conjunctive Queries. *Journal of the ACM*, 59(6):1–87, 2012. ISSN 00045411. doi: 10.1145/2395116.2395119. URL http://dl.acm.org/citation.cfm?doid=2395116.2395119.

D'Amato, Claudia, Fanizzi, Nicola, and Lukasiewicz, Thomas. Tractable reasoning with bayesian description logics. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5291 LNAI(iii):146–159, 2008. ISSN 03029743. doi: 10.1007/978-3-540-87993-0_13.

De Raedt, Luc, Kimmig, Angelika, and Toivonen, Hannu. ProbLog: A Probabilistic Prolog and Its Application in Link Discovery. In Sangal, Rajeev, Mehta, Harish, and Bagga, R K, editors, *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, volume 7, pages 2462–2467, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc. URL http://www.aaai.org/Papers/IJCAI/2007/IJCAI07-397.pdf.

Domingos, Pedro M and Webb, William Austin. A Tractable First-Order Probabilistic Logic. In *AAAI*, 2012.

Dong, Xin, Gabrilovich, Evgeniy, Heitz, Geremy, Horn, Wilko, Lao, Ni, Murphy, Kevin, Strohmann, Thomas, Sun, Shaohua, and Zhang, Wei. Knowledge vault: a web-scale approach to probabilistic knowledge fusion. *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '14*, pages 601–610, 2014a. ISSN 0893-6080. doi: 10.1145/2623330.2623623. URL http://dl.acm.org/citation.cfm?doid=2623330.2623623.

Dong, Xin Luna, Gabrilovich, Evgeniy, Heitz, Geremy, Horn, Wilko, Lao, Ni, Murphy, Kevin, Strohmann, Thomas, Sun, Shaohua, and Zhang, Wei. Knowledge Vault: A Web-Scale Approach to Probabilistic Knowledge Fusion. *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 601–610, 2014b. doi: 10.1145/2623330.2623623.

Dubois, Didier and Prade, Henri. Possibility theory, probability theory and multiple-valued logics: A clarification. *Annals of Mathematics and Artificial Intelligence*, 32 (1-4):35–66, 2001. ISSN 10122443. doi: 10.1023/A:1016740830286.

Duerst, Martin and Suignard, Michel. Internationalized Resource Identifiers (IRIs). RFC 3987 (Proposed Standard), 2005. ISSN 2070-1721. URL `https://www.rfc-editor.org/rfc/rfc3987.txt`.

Dylla, Maximilian, Sozio, Mauro, and Theobald, Martin. Resolving Temporal Conflicts in Inconsistent RDF Knowledge Bases. In *14. GI-Fachtagung Datenbanksysteme für Business, Technologie und Web (BTW)*, pages 474–493, 2011.

Dylla, Maximilian, Miliaraki, Iris, and Theobald, Martin. A temporal-probabilistic database model for information extraction. *Proceedings of the VLDB Endowment*, 6(14):1810–1821, 2013. URL `http://dl.acm.org/citation.cfm?id=2556564`.

Ekelhart, Andreas, Fenz, Stefan, Klemen, Markus D., and Weippl, Edgar R. Security Ontology : Simulating Threats to Corporate Assets. *Proceedings of the 2nd Intl. Conf. on Information Systems Security (ICISS)*, pages 249–259, 2006. doi: http://dx.doi.org/10.1007/11961635_17.

Ernst & Young. Ad-hoc information security solutions no longer an option, as companies struggle to keep pace with today's threats, 2012. URL `http://www.ey.com/ru/en/newsroom/news-releases/press-release---2012-10-29-2`.

Ernst & Young. Managing IT risk in a fast-changing environment. Technical report, Ernst & Young, 2013.

Etzioni, Oren, Banko, Michele, Soderland, Stephen, and Weld, Daniel S. Open information extraction from the web. *Communications of the ACM*, 51(12):68–74, 2008.

European Union Agency for Network and Information Security. ENISA Threat Landscape mid year 2013. Technical report, European Union Agency for Network and Information Security, 2013. URL `http://www.enisa.europa.eu/activities/risk-management/evolving-threat-environment/enisa-threat-landscape-mid-year-2013`.

Fader, Anthony, Soderland, Stephen, and Etzioni, Oren. Identifying Relations for Open Information Extraction. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545, 2011. ISSN 1937284115. doi: 10.1234/12345678. URL `http://dl.acm.org/citation.cfm?id=2145432.2145596%5Cnhttp://dl.acm.org/citation.cfm?id=2145596`.

Fionda, Valeria and Pirrò, Giuseppe. Querying graphs with preferences. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 929–938. ACM, 2013.

Fleischhacker, Daniel. Repairing learned ontologies. In *CEUR Workshop Proceedings*, volume 1162, pages 15–26, 2014.

Galárraga, Luis, Teflioudi, Christina, Hose, Katja, and Suchanek, Fabian M. Fast rule mining in ontological knowledge bases with AMIE+. *VLDB Journal*, 24(6):707–730, 2015. ISSN 0949877X. doi: 10.1007/s00778-015-0394-1.

Gallego, Mario Arias Fernández, Javier D. Martínez-Prieto, Miguel A. and de la Fuente, Pablo. An Empirical Study of Real-World SPARQL Queries. In *1st International Workshop on Usage Analysis and the Web of Data (USEWOD2011) at the 20th International World Wide Web Conference (WWW 2011)*, pages 10–13, 2011. doi: 10.1016/j.postharvbio.2004.03.006. URL `http://arxiv.org/abs/1103.5043`.

Gerla, Giangiacomo. Inferences in probability logic. *Artificial Intelligence*, 70(1-2):33–52, 1994. ISSN 00043702. doi: 10.1016/0004-3702(94)90102-3.

Glimm, Birte, Horrocks, Ian, Motik, Boris, Stoilos, Giorgos, and Wang, Zhe. HermiT: An OWL 2 Reasoner. *Journal of Automated Reasoning*, 53(3):245–269, 2014. ISSN 15730670. doi: 10.1007/s10817-014-9305-1.

Gray, J and Siewiorek, D P. High-availability computer systems. *Computer*, 24(9):39–48, 1991. ISSN 0018-9162. doi: 10.1109/2.84898.

Guarino, Nicola and Welty, Christopher A. An Overview of OntoClean. *Handbook on Ontologies*, pages 1–20, 2009. URL `http://link.springer.com/chapter/10.1007/978-3-540-92673-3_9`.

Guha, Ramanathan V, Brickley, Dan, and Macbeth, Steve. Schema. org: Evolution of structured data on the web. *Communications of the ACM*, 59(2):44–51, 2016.

Guo, Yuanbo, Pan, Zhengxiang, and Heflin, Jeff. LUBM: A Benchmark for OWL Knowledge Base Systems. In *Web Semantics*, volume 3, pages 158–182, 2005. ISBN 1570-8268. doi: 10.1016/j.websem.2005.06.005.

Gutierrez, Claudio, Hurtado, Carlos, and Vaisman, Alejandro. Temporal RDF. *The Semantic Web: Research and Applications*, pages 93–107, 2005. ISSN 1539-4794. doi: 10.1007/11431053_7. URL `http://link.springer.com/chapter/10.1007/11431053_7`.

Han, Xingwang, Feng, Zhiyong, Zhang, Xiaowang, Wang, Xin, Rao, Guozheng, and Jiang, Shuo. On the Statistical Analysis of Practical SPARQL Queries. *arXiv preprint arXiv:1603.06729*, page 6, 2016. URL `http://arxiv.org/abs/1603.06729`.

Heiden, Eric, Bader, Sebastian, and Kirste, Thomas. Concept and Realization of a Diagnostic System for Smart Environments. In van den Herik, H Jaap, Rocha, Ana Paula, and Filipe, Joaquim, editors, *Proceedings of the 9th International Conference on Agents and Artificial Intelligence, ICAART 2017, Volume 2, Porto, Portugal, February 24-26, 2017.*, pages 318–329. SciTePress, 2017. ISBN 978-989-758-220-2. doi: 10.5220/0006257903180329. URL `https://doi.org/10.5220/0006257903180329`.

Hess, Claudia, Chen, Willy, and Syldatke, Thomas. Business-oriented CAx integration with semantic technologies revisited. In *INFORMATIK 2010 - Service Science - Neue Perspektiven fur die Informatik, Beitrage der 40. Jahrestagung der Gesellschaft fur Informatik e.V. (GI)*, volume 2, pages 115–120, 2010. ISBN 9783885792703. URL `http://www.scopus.com/inward/record.url?eid=2-s2.0-84874273263&partnerID=40&md5=7ac60c0bb767d63a7ff33c827abb4f1c`.

Hilber, David and Ackermann, Wilhelm. *Grundzüge der theoretischen Logik*. Number 27. Springer, Berlin, zweite auf edition, 1938. ISBN 9783662417805. doi: 10.1007/978-3-662-41928-1.

Hoffart, Johannes, Suchanek, Fabian M, Berberich, Klaus, and Weikum, Gerhard. YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia. *Artificial Intelligence*, 194:28–61, 2013.

Huang, Jiewen, Antova, Lyublena, Koch, Christoph, and Olteanu, Dan. MayBMS: A Probabilistic Database Management System. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pages 1071–1074. ACM Press, 2009. ISBN 9781605585512. doi: 10.1145/1559845.1559984. URL `http://dl.acm.org/citation.cfm?id=1559984http://portal.acm.org/citation.cfm?doid=1559845.1559984`.

Huber, Jakob, Meilicke, Christian, and Stuckenschmidt, Heiner. Applying Markov Logic for Debugging Probabilistic Temporal Knowledge Bases. In *Proceedings of the 4th Workshop on Automated Knowledge Base Construction (AKBC)*, 2014.

IBM X-Force. Mid-Year Trend and Risk Report 2013. Technical report, IBM X-Force, 2013. URL `http://www-03.ibm.com/security/xforce/downloads.html`.

Inoue, Naoya and Inui, Kentaro. ILP-Based Reasoning for Weighted Abduction. *Plan, Activity, and Intent Recognition*, pages 25–32, 2011. URL `http://www.aaai.org/ocs/index.php/WS/AAAIW11/paper/download/3999/4313`.

Jain, Dominik. Knowledge Engineering with Markov Logic Networks: A Review. In Beierle, C and Kern-Isberner, G, editors, *Evolving Knowledge in Theory and Applications. 3rd Workshop on Dynamics of Knowledge and Belief (DKB-2011) at the 34th Annual German Conference on Artificial Intelligence, KI-2011, Berlin, Germany, October 4, 2011. Proceedings*, volume 361 of *Informatik-Bericht*, pages 16–30. Fakultät für Mathematik und Informatik, FernUniversität in Hagen, 2011. URL `http://www.fernuniversitthagen.de/wbs/research/papers/res/BeierleKernIsberner2011InfBericht361.pdf#page=24`.

Jain, Dominik, Kirchlechner, Bernhard, and Beetz, Michael. Extending Markov Logic to Model Probability Distributions in Relational Domains. In Hertzberg, Joachim, Beetz, Michael, and Englert, Roman, editors, *KI 2007: Advances in Artificial Intelligence*, volume 4667 of *Lecture Notes in Computer Science*, pages 129–143. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-74564-8. doi: 10.1007/

978-3-540-74565-5_12. URL http://dx.doi.org/10.1007/978-3-540-74565-5_12http://link.springer.com/chapter/10.1007/978-3-540-74565-5_12.

Jensen, Finn V. *An Introduction to Bayesian Networks*, volume 210. UCL press London, 1996.

Jha, Abhay and Suciu, Dan. Probabilistic Databases with MarkoViews. *Proceedings of the VLDB Endowment*, 5(11):1160–1171, 2012. URL http://dl.acm.org/citation.cfm?id=2350236.

Jobczyk, Krystian. Towards a project of fuzzy logic of real analysis. 2016.

Jones, Dean, Bench-Capon, Trevor, and Visser, Pepin. Methodologies for ontology development. In *Proceedings of the 15th IFIP World Computer Congress*, pages 20–35, 1998. ISBN 3-85403-122-X. doi: 10.1.1.52.2437. URL http://www.springerlink.com/index/u3u53033q2508524.pdf%5Cnhttp://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.52.2437&rep=rep1&type=pdf.

Jøsang, Audun. A Logic for Uncertain Probabilities. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 09(03):279–311, 2001. ISSN 0218-4885. doi: 10.1142/S0218488501000831. URL http://www.worldscientific.com/doi/abs/10.1142/S0218488501000831.

Jung, Jean Christoph and Lutz, Carsten. Ontology-Based Access to Probabilistic Data with OWL QL. In *The Semantic Web - ISWC 2012*, pages 182–197. Springer, 2012.

Kakas, Antonis C., Kowalski, Robert A., and Toni, Francesca. Abductive logic programming. *J. Log. Comput.*, 1(6):719–770, 1992. URL http://logcom.oxfordjournals.org/content/2/6/719.short.

Kalyanpur, Aditya, Boguraev, Branimir K., Patwardhan, Siddharth, Murdock, J. William, Lally, Adam, Welty, Christopher A., Prager, John M., Coppola, Bonaventura, Fokoue-Nkoutche, Achille, Zhang, Lei, Pan, Yue, and Qui, Zhao Ming. Structured Data and Inference in DeepQA. *IBM Journal of Research and Development*, 56(3):351–364, 2012. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6177725.

Kandula, Srikanth, Katabi, Dina, and Vasseur, JP P. Shrink: A tool for failure diagnosis in IP networks. *Proceedings of the 2005 ACM SIGCOMM workshop on Mining network data*, pages 173–178, 2005. URL http://dl.acm.org/citation.cfm?id=1080178.

Kaplan, Stanley and Garrick, B John. On The Quantitative Definition of Risk. *Risk Analysis*, 1(1):11–27, 1981. ISSN 1539-6924. doi: 10.1111/j.1539-6924.1981.tb01350.x. URL http://dx.doi.org/10.1111/j.1539-6924.1981.tb01350.x.

Kate, Rohit J and Mooney, Raymond J. Probabilistic Abduction using Markov Logic Networks. *IJCAI-09 Workshop on Plan, Activity, and Intent Recognition*, 2009. URL http://userweb.cs.utexas.edu/users/ml/papers/kate-pair09.pdf.

Kazakov, Yevgeny. Consequence-Driven Reasoning for Horn SHIQ Ontologies. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence, IJCAI 2009*, pages 2040–2045, Pasadena, California, USA, 2009.

Kersting, Kristian and De Raedt, Luc. Bayesian Logic Programs. *CoRR*, cs.AI/0111, 2001.

Kimmig, Angelika, Mihalkova, Lilyana, and Getoor, Lise. Lifted graphical models: a survey. *Machine Learning*, 99(1):1–45, 2014. ISSN 1573-0565. doi: 10.1007/s10994-014-5443-2. URL `http://dx.doi.org/10.1007/s10994-014-5443-2`.

Klinov, Pavel and Parsia, Bijan. Optimization and Evaluation of Reasoning in Probabilistic Description Logic: Towards a Systematic Approach. In *International Semantic Web Conference*, pages 213–228, 2008.

Klinov, Pavel and Parsia, Bijan. Pronto: A Practical Probabilistic Description Logic Reasoner. *Lecture Notes in Computer Science*, 7123:59–79, 2013. ISSN 03029743. doi: 10.1007/978-3-642-35975-0-4.

Klir, George J and Yuan, Bo. *Fuzzy Sets and Fuzzy Logic: Theory and Applications.* 1995. ISBN 0131011715. doi: 10.1109/9780470544341.ch11.

Koller, Daphne and Friedman, Nir. *Probabilistic Graphical Models: Principles and Techniques*, volume 2009. 2009. ISBN 0262013193. doi: 10.1016/j.ccl.2010.07.006. URL `http://mitpress.mit.edu/catalog/item/default.asp?ttype=2&tid=11886`.

Koller, Daphne, Levy, Alon, and Pfeffer, Avi. P-CLASSIC: A tractable probablistic description logic. *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, (August):390–397, 1997.

Krötzsch, Markus. Efficient Inferencing for OWL EL. In *Logics in Artificial Intelligence - 12th European Conference, JELIA 2010, Helsinki, Finland, September 13-15, 2010. Proceedings*, volume 6341 of *Lecture Notes in Computer Science*, pages 234–246. Springer, 2010.

Lakshmanan, Laks V. S. and Sadri, Fereidoon. Probabilistic Deductive Databases. *SLP*, pages 254–268, jun 1994. ISSN 1097-6809. doi: 10.1016/j.jvs.2014.03.253.

Lanti, Davide, Rezk, Martin, Xiao, Guohui, and Calvanese, Diego. The NPD Benchmark: Reality Check for OBDA Systems. *Proc. of the 18th Int. Conf. on Extending Database Technology (EDBT)*, 2015.

Lehmann, Jens, Isele, Robert, Jakob, Max, Jentzsch, Anja, Kontokostas, Dimitris, Mendes, Pablo N., Hellmann, Sebastian, Morsey, Mohamed, Van Kleef, Patrick, Auer, Sören, and Bizer, Christian. DBpedia - A large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web*, 6(2):167–195, 2015. ISSN 22104968. doi: 10.3233/SW-140134.

Ling, Xiao and Weld, Daniel S. Temporal Information Extraction. *Artificial Intelligence*, pages 1385–1390, 2007. URL `http://www.cs.washington.edu/homes/weld/papers/ling-aaai10.pdf`.

Ling, Xiao and Weld, Daniel S. Temporal Information Extraction. In *AAAI*, 2010.

Liu, T.S. and Chiou, S.B. The application of Petri nets to failure analysis. *Reliability Engineering and System Safety*, 57(2):129–142, 1997. ISSN 09518320. doi: 10.1016/S0951-8320(97)00030-6. URL `http://linkinghub.elsevier.com/retrieve/pii/S0951832097000306`.

Lukasiewicz, Thomas. Probabilistic Logic Programming. *13th European Conference on Artificial Intelligence (ECAI-98)*, (July):388–392, 1998.

Lukasiewicz, Thomas. Probabilistic description logic programs. *International Journal of Approximate Reasoning*, 45(2):288–307, jul 2007. ISSN 0888613X. doi: 10.1016/j.ijar.2006.06.012. URL `http://linkinghub.elsevier.com/retrieve/pii/S0888613X06000648`.

Lukasiewicz, Thomas. Expressive probabilistic description logics. *Artificial Intelligence*, 172(6-7):852–883, apr 2008. ISSN 00043702. doi: 10.1016/j.artint.2007.10.017. URL `http://linkinghub.elsevier.com/retrieve/pii/S0004370207001877`.

Lukasiewicz, Thomas and Straccia, Umberto. Managing Uncertainty and Vagueness in Description Logics for the Semantic Web. *Web Semantics*, 6(4):291–308, 2008. ISSN 15708268. doi: 10.1016/j.websem.2008.04.001.

Lutz, Carsten, Seylan, Inanç, Toman, David, and Wolter, Frank. The Combined Approach to OBDA: Taming Role Hierarchies Using Filters. *Lecture Notes in Computer Science*, 8218 LNCS:314–330, 2013. ISSN 03029743. doi: 10.1007/978-3-642-41335-3_20.

Marwede, Nina, Rohr, Matthias, and Hasselbring, Wilhelm. Automatic Failure Diagnosis Support in Distributed Large-Scale Software Systems based on Timing Behavior Anomaly Correlation. In Winter, A., Ferenc, R., and Kodel, J., editors, *Proceeding of the 13th European Conference on Software Maintenance and Reengineering (CSMR 2009)*, pages 47–57, 2009.

Mikosch, Thomas and Kallenberg, Olav. Foundations of Modern Probability. *Journal of the American Statistical Association*, 93(443):1243, 1998. ISSN 01621459. doi: 10.2307/2669881. URL `http://www.jstor.org/stable/2669881?origin=crossref`.

Mohammadi, Dara. Finding patient zero. *Pharmaceutical Journal*, 294(7845):47–49, 2015. ISSN 00316873.

Motik, Boris. Representing and querying validity time in RDF and OWL: A logic-based approach. *Web Semantics: Science, Services and Agents on the World Wide Web*, 12:3–21, 2012.

Musen, Mark A. The Protégé Project: A Look Back and a Look Forward. *AI Matters*, 1(4):4–12, 2015. doi: 10.1145/2757001.2757003.

Ng, Hwee Tou and Mooney, Raymond J. An Efficient First-Order Abduction System Based on the ATMS. In *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91)*, pages 494–499, Anaheim, CA, 1991. University of Texas at Austin.

Ng, Raymond and Subrahmanian, V.S. Probabilistic logic programming. *Information and Computation*, 101(2):150–201, 1992. ISSN 08905401. doi: 10.1016/0890-5401(92)90061-J. URL `http://linkinghub.elsevier.com/retrieve/pii/089054019290061J%5Cnhttp://www.sciencedirect.com/science/article/pii/089054019290061J`.

Niepert, Mathias, Noessner, Jan, Meilicke, Christian, and Stuckenschmidt, Heiner. Probabilistic-Logical Web Data Integration. In Polleres, Axel, D Amato, Claudia, Arenas, Marcelo, Handschuh, Siegfried, Kroner, Paula, Ossowski, Sascha, and Patel-Schneider, Peter, editors, *Reasoning Web. Semantic Technologies for the Web of Data*, volume 6848 of *Lecture Notes in Computer Science*, pages 504–533. Springer Berlin Heidelberg, 2011a. ISBN 978-3-642-23031-8. doi: 10.1007/978-3-642-23032-5_11. URL `http://dx.doi.org/10.1007/978-3-642-23032-5_11`.

Niepert, Mathias, Noessner, Jan, and Stuckenschmidt, Heiner. Log-Linear Description Logics. In *IJCAI International Joint Conference on Artificial Intelligence*, pages 2153–2158, 2011b. ISBN 9781577355120. doi: 10.5591/978-1-57735-516-8/IJCAI11-359.

Nilsson, Nils J. Probabilistic logic. *Artificial Intelligence*, 28(1):71–87, 1986. ISSN 00043702. doi: 10.1016/0004-3702(86)90031-7.

Noessner, Jan, Niepert, Mathias, and Stuckenschmidt, Heiner. RockIt: Exploiting Parallelism and Symmetry for MAP Inference in Statistical Relational Models. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, pages 739–745, Bellevue, Washington, USA, 2013. AAAI Press. URL `http://link.springer.com/chapter/10.1007/BFb0027523`.

Ovchinnikova, Ekaterina, Montazeri, Niloofar, Alexandrov, Theodore, R., Hobbs Jerry, Mccord, Michael C., and Mulkar-Mehta, Rutu. Abductive Reasoning with a Large Knowledge Base for Discourse Processing. In Bunt, Harry, Bos, Johan, and Pulman, Stephen, editors, *Computing Meaning: Volume 4*, pages 107–127. Springer Netherlands, Dordrecht, 2014. ISBN 978-94-007-7284-7. doi: 10.1007/978-94-007-7284-7_7. URL `http://dx.doi.org/10.1007/978-94-007-7284-7_7`.

Papaioannou, Katerina and Bohlen, Michael. TemProRA: Top-k temporal-probabilistic results analysis. In *2016 IEEE 32nd International Conference on Data Engineering, ICDE 2016*, pages 1382–1385, 2016. ISBN 9781509020195. doi: 10.1109/ICDE.2016.7498350.

Parsia, Bijan, Patel-Schneider, Peter, Hitzler, Pascal, Rudolph, Sebastian, and Krötzsch, Markus. OWL 2 Web Ontology Language Primer (Second Edition). Technical report, W3C, 2012. URL `https://www.w3.org/TR/owl2-primer/`.

Paulheim, Heiko and Pan, Jeff Z. Why the Semantic Web Should Become More Imprecise. In *In Proceedings of What Will the Semantic Web Look Like 10 Years From Now?*, pages 1–5, 2012. URL `http://www.heikopaulheim.com/docs/sw2020.pdf`.

Pearl, Judea. *Probabilistic Reasoning in Intelligent Systems.* Morgan Kaufmann Publishers Inc., 1988.

Picalausa, Francois and Vansummeren, Stijn. What are Real SPARQL Queries Like? In *Proceedings of the International Workshop on Semantic Web Information Management*, pages 1–6, 2011. ISBN 9781450306515. doi: 10.1145/1999299.1999306. URL `http://portal.acm.org/citation.cfm?doid=1999299.1999306`.

Poole, David. The independent choice logic for modelling multiple agents under uncertainty. *Artificial Intelligence*, 94(1-2):7–56, 1997. ISSN 00043702. doi: 10.1016/S0004-3702(97)00027-1. URL `http://www.sciencedirect.com/science/article/pii/S0004370297000271`.

Poole, David. The independent choice logic and beyond. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 4911 LNAI:222–243, 2008. ISSN 03029743. doi: 10.1007/978-3-540-78652-8_8.

Poole, David L., Goebel, Randy G., and Aleliunas, Romas. Theorist: A logical reasoning system for defaults and diagnosis. In Cercone, Nick J. and McCalla, Gordon, editors, *The Knowledge Frontier: Essays in the Representation of Knowledge*, pages 331–352. Springer, 1987.

Poon, Hoifung and Vanderwende, Lucy. Joint inference for knowledge extraction from biomedical literature. *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, (June):813–821, 2010. URL `http://portal.acm.org/citation.cfm?id=1858122%5Cnhttp://dl.acm.org/citation.cfm?id=1858122`.

Poon, Hoifung, Domingos, Pedro, and Sumner, Marc. A General Method for Reducing the Complexity of Relational Inference And its Application to MCMC. *Proceedings of the 23rd National Conference on Artificial Intelligence (AAAI)*, 2:1075—-1080, 2008. ISSN 00219258. doi: 10.1074/jbc.M201091200.

Pople, Harry E. On the Mechanization of Abductive Logic. *IJCAI*, pages 147–152, 1973. URL `http://pdf.aminer.org/000/366/748/two_sided_hypotheses_generation_for_abductive_analogical_reasoning.pdf`.

Pujara, Jay, Miao, Hui, Getoor, Lise, and Cohen, William. Knowledge graph identification. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 8218 LNCS, pages 542–557, 2013. ISBN 9783642413346. doi: 10.1007/978-3-642-41335-3_34.

Raghavan, Sindhu and Mooney, Raymond J. Bayesian Abductive Logic Programs. *Stat. Relational Artif. Intell.*, pages 82–87, 2010. URL `http://www.aaai.org/ocs/index.php/WS/AAAIW10/paper/viewPDFInterstitial/1974/2487`.

Richardson, Matthew and Domingos, Pedro. Markov Logic Networks. *Machine Learning*, 62(1-2):107–136, jan 2006. ISSN 0885-6125. doi: 10.1007/s10994-006-5833-1. URL `http://dx.doi.org/10.1007/s10994-006-5833-1`.

Riguzzi, Fabrizio, Bellodi, Elena, Lamma, Evelina, and Zese, Riccardo. BUNDLE: A Reasoner for Probabilistic Ontologies. In *Reasoning and Rule Systems - 7th International Conference, RR 2013*, Lecture Notes in Computer Science, pages 183–197, Mannheim, Germany,, 2013. Springer. URL `http://link.springer.com/chapter/10.1007/978-3-642-39666-3_14`.

Riguzzi, Fabrizio, Bellodi, Elena, Lamma, Evelina, and Zese, Riccardo. Probabilistic Description Logics under the Distribution Semantics. *The Semantic Web*, 6(5):477–501, 2015. doi: 10.3233/SW-140154. URL `http://www.semantic-web-journal.net/system/files/swj651.pdf`.

Ritze, Dominique. *Web-Scale Web Table to Knowledge Base Matching.* PhD thesis, 2017.

Ritze, Dominique, Lehmberg, Oliver, Oulabi, Yaser, and Bizer, Christian. Profiling the Potential of Web Tables for Augmenting Cross-domain Knowledge Bases Categories and Subject Descriptors. *Www*, pages 251–261, 2016. doi: 10.1145/2872427.2883017. URL `http://www2016.net/proceedings/proceedings/p251.pdf`.

Rooney, J.J. and Heuvel, L.N.V. Root cause analysis for beginners. *Quality Progress*, 37(7):45–53, 2004. URL `https://servicelink.pinnacol.com/pinnacol_docs/lp/cdrom_web/safety/management/accident_investigation/Root_Cause.pdf`.

Ruspini, Enrique H., Lowrance, John D., and Strat, Thomas M. Understanding evidential reasoning. *International Journal of Approximate Reasoning*, 6(3):401–424, 1992. ISSN 0888613X. doi: 10.1016/0888-613X(92)90033-V.

Russell, Stuart J., Norvig, Peter, and By, Uploaded. *Artificial Intelligence: A Modern Approach.* 2010. ISBN 0137903952. doi: 10.1017/S0269888900007724. URL `http://amazon.de/o/ASIN/0130803022/`.

Saleem, Muhammad, Ali, Muhammad Intizar, Hogan, Aidan, Mehmood, Qaiser, and Ngonga Ngomo, Axel Cyrille. LSQ: The Linked SPARQL Queries Dataset. *Lecture Notes in Computer Science*, 9367:261–269, 2015. ISSN 16113349. doi: 10.1007/978-3-319-25010-6_15.

Sato, Taisuke. A statistical learning method for logic programs with distribution semantics. In *Proceedings of the Twelth International conference on logic programming*, pages 715–729, 1995. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.17.4408`.

Sato, Taisuke and Kameya, Yoshitaka. New advances in logic-based probabilistic modeling by PRISM. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 4911 LNAI:118–155, 2008. ISSN 03029743. doi: 10.1007/978-3-540-78652-8_5.

Schlobach, Stefan, Huang, Zhisheng, Cornet, Ronald, and van Harmelen, Frank. Debugging Incoherent Terminologies. *Journal of Automated Reasoning*, 39(3):317–349, 2007. doi: 10.1007/s10817-007-9076-z.

Schmidt, Michael, Hornung, Thomas, Lausen, Georg, and Pinkel, Christoph. SP$^2$Bench: A SPARQL Performance Benchmark. *Data Engineering*, 25:222–23, 2009. ISSN 1084-4627. doi: 10.1109/ICDE.2009.28. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4812405`.

Schoenfisch, J. Querying probabilistic ontologies with SPARQL. In *Lecture Notes in Informatics (LNI), Proceedings - Series of the Gesellschaft fur Informatik (GI)*, volume P-232, 2014. ISBN 9783885796268.

Schoenfisch, Joerg and Stuckenschmidt, Heiner. Towards Large-Scale Probabilistic OBDA. In Beierle, Christoph and Dekhtyar, Alex, editors, *Scalable Uncertainty Management: 9th International Conference, SUM 2015, Québec City, QC, Canada, September 16-18, 2015. Proceedings*, pages 106–120. Springer International Publishing, Cham, 2015. ISBN 978-3-319-23540-0. doi: 10.1007/978-3-319-23540-0_8.

Schoenfisch, Joerg and Stuckenschmidt, Heiner. Analyzing Real-World SPARQL Queries in the Light of Probabilistic Data. *Proceedings of the 12th International Workshop on Uncertainty Reasoning for the Semantic Web (URSW 2016)*, 1665:13–23, 2016. URL `http://ceur-ws.org/Vol-1665/paper2.pdf`.

Schoenfisch, Joerg and Stuckenschmidt, Heiner. Analyzing Real-World SPARQL Queries and Ontology-Based Data Access in the Context of Probabilistic Data. *International Journal of Approximate Reasoning J. Schoenfisch, H. Stuckenschmidt Int. J. Approx. Reason*, 90:374–388, 2017. ISSN 0888613X. doi: 10.1016/j.ijar.2017.08.005. URL `http://dx.doi.org/10.1016/j.ijar.2017.08.005`.

Schoenfisch, Joerg, von Stülpnagel, Janno, Ortmann, Jens, Meilicke, Christian, and Stuckenschmidt, Heiner. Root Cause Analysis through Abduction in Markov Logic Networks. In *Proceedings of the 20th International Enterprise Distributed Object Computing Conference (EDOC)*, pages 1–13, 2016.

Schoenfisch, Joerg, Meilicke, Christian, von Stülpnagel, Janno, Ortmann, Jens, and Stuckenschmidt, Heiner. Root Cause Analysis in IT Infrastructures Using Ontologies and Abduction in Markov Logic Networks. *Information Systems*, 73, 2017. doi: 10.1016/j.is.2017.11.003.

Schoenmackers, Stefan, Etzioni, Oren, and Weld, Daniel S. Scaling textual inference to the web. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 59(October):79, 2008. doi: 10.3115/1613715.1613727. URL `http://portal.acm.org/citation.cfm?doid=1613715.1613727`.

Schoenmackers, Stefan (University of Washington/Turing Center), Etzioni, Oren (University of Washington/Turing Center), Weld, Daniel S. (University of Washington/Turing Center), and Davis, Jesse (Katholieke Universiteit Leuven). Learning first-order horn clauses from web text, 2010. URL `http://dl.acm.org/citation.cfm?id=1870764`.

Schreiber, Guus and Raimond, Yves. RDF 1.1 Primer. W3C note, W3C, 2014. URL `http://www.w3.org/TR/2014/NOTE-rdf11-primer-20140624/`.

Simancik, Frantisek, Kazakov, Yevgeny, and Horrocks, Ian. Consequence-Based Reasoning beyond Horn Ontologies. In *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, pages 1093–1098. IJCAI/AAAI, 2011.

Singh, Sameer, Wick, Michael, and McCallum, A. Monte Carlo MCMC: efficient inference by approximate sampling. *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 1104—-1113, 2012. doi: 10.1007/978-3-642-31164-2. URL `http://dl.acm.org/citation.cfm?id=2391072`.

Singla, Parag and Domingos, Pedro. Lifted First-Order Belief Propagation. In *Proceedings of AAAI*, pages 1094–1099, 2008.

Singla, Parag and Mooney, Raymond J. Abductive Markov Logic for Plan Recognition. *AAAI*, pages 1069–1075, 2011. URL `http://www.aaai.org/ocs/index.php/AAAI/AAAI11/paper/viewPDFInterstitial/3615/3992`.

Sirin, Evren, Parsia, Bijan, Grau, Bernardo Cuenca, Kalyanpur, Aditya, and Katz, Yarden. Pellet: A practical OWL-DL reasoner. *Web Semantics*, 5(2):51–53, 2007. ISSN 15708268. doi: 10.1016/j.websem.2007.03.004.

Steinder, M. and Sethi, A.S. S. Increasing robustness of fault localization through analysis of lost, spurious, and positive symptoms. *Proceedings of the Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2002)*, 1:322–331, 2002. doi: 10.1109/INFCOM.2002.1019274. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1019274`.

Stickel, Mark E. A Prolog-like inference system for computing minimum-cost abductive explanations in natural-language interpretation. *Annals of Mathematics and Artificial*

*Intelligence*, 4(1-2):89–105, 1991. URL `http://link.springer.com/article/10.1007/BF01531174`.

Suciu, Dan, Olteanu, Dan, Ré, Christopher, and Koch, Christoph. *Probabilistic Databases*. Morgan & Claypool Publishers, 2011. ISBN 9781608456802.

Tansel, Abdullah Uz, Clifford, James, Gadia, Shashi, Jajodia, Sushil, Segev, Arie, and Snodgrass, Richard, editors. *Temporal Databases: Theory, Design, and Implementation*. Benjamin-Cummings Publishing Co., Inc., Redwood City, CA, USA, 1993. ISBN 0-8053-2413-5.

The W3C SPARQL Working Group. SPARQL 1.1 Overview. W3C recommendation, W3C, 2013. URL `https://www.w3.org/TR/sparql11-overview/`.

Theobald, Martin, De Raedt, Luc, Dylla, Maximilian, Kimmig, Angelika, and Miliaraki, Iris. 10 Years of Probabilistic Querying - What Next? *Advances in Databases and Information Systems*, pages 1–13, 2013. URL `https://lirias.kuleuven.be/handle/123456789/403578`.

Venugopal, Deepak and Gogate, Vibhav. On Lifting the Gibbs Sampling Algorithm. In *Proceedings of NIPS*, pages 1664–1672. 2012.

vom Brocke, Jan, Braccini, Alessio Maria, Sonnenberg, Christian, and Spagnoletti, Paolo. Living IT infrastructures - An ontology-based approach to aligning IT infrastructure capacity and business needs. *International Journal of Accounting Information Systems*, 15(3):246–274, 2014. ISSN 14670895. doi: 10.1016/j.accinf.2013.10.004.

von Stülpnagel, Janno, Ortmann, Jens, and Schoenfisch, Joerg. IT Risk Management with Markov Logic Networks. *Advanced Information Systems Engineering*, pages 301—-315, 2014. URL `http://link.springer.com/chapter/10.1007/978-3-319-07881-6_21`.

Wang, Daisy Zhe, Michelakis, Eirinaios, Garofalakisy, Minos, and Hellerstein, Joseph M. BayesStore: Managing Large, Uncertain Data Repositories with Probabilistic Graphical Models. In *Proceedings of the VLDB Endowment*, volume 1, pages 340–351, 2008. ISBN 0000000000000. doi: 10.1145/1453856.1453896. URL `http://www.scopus.com/inward/record.url?eid=2-s2.0-84859207862&partnerID=40&md5=8c49b689d426d4f9a27f064594eba37c`.

Weber, P, Medina-Oliva, G, Simon, C, and Iung, B. Overview on Bayesian networks applications for dependability, risk analysis and maintenance areas. *Engineering Applications of Artificial Intelligence*, 25(4):671–682, 2012. ISSN 0952-1976. doi: http://dx.doi.org/10.1016/j.engappai.2010.06.002. URL `http://www.sciencedirect.com/science/article/pii/S095219761000117X`.

Weidl, G., Madsen, A. L., and Israelson, S. Applications of object-oriented Bayesian networks for condition monitoring, root cause analysis and decision support on operation of complex continuous processes. *Computers and Chemical Engineering*, 29(9): 1996–2009, 2005. ISSN 00981354. doi: 10.1016/j.compchemeng.2005.05.005.

Widom, Jennifer. Trio: A System for Integrated Management of Data, Accuracy, and Lineage. *Technical Report*, pages 1–22, 2004. URL `http://ilpubs.stanford.edu:8090/658`.

Wu, Wentao, Li, Hongsong, Wang, Haixun, and Zhu, Kenny Q. Probase: A probabilistic taxonomy for text understanding. *Proceedings of the 2012 ACM SIGMOD*, pages 481–492, 2012. ISSN 00043702. doi: 10.1016/j.artint.2011.01.003. URL `http://dl.acm.org/citation.cfm?id=2213891`.

Zadeh, Lotfi A. Fuzzy sets. *Information and Control*, 8(3):338–353, 1965. ISSN 0019-9958. doi: 10.1016/S0019-9958(65)90241-X.

Zambon, Emmanuele, Etalle, Sandro, Wieringa, Roel J., and Hartel, Pieter. Architecture-based Qualitative Risk Analysis for Availability of IT Infrastructures. Technical report, Centre for Telematics and Information Technology, University of Twente, 2009.

Zawawy, Hamzeh, Kontogiannis, Kostas, Mylopoulos, John, and Mankovskii, Serge. Requirements-driven root cause analysis using markov logic networks. In Ralyte, Jolita, Franch, Xavier, Brinkkemper, Sjaak, and Wrycza, Stanislaw, editors, *Advanced Information Systems Engineering*, volume 7328 of *Lecture Notes in Computer Science*, pages 350–365. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-31094-2. doi: 10.1007/978-3-642-31095-9_23. URL `http://dx.doi.org/10.1007/978-3-642-31095-9_23`.

Zhou, Xiaofeng, Chen, Yang, and Wang, Daisy Zhe. ArchimedesOne : Query Processing over Probabilistic Knowledge Bases. *Vldb*, pages 5–8, 2016. ISSN 2150-8097. doi: 10.14778/3007263.3007284.