

UNIVERSITY OF SÃO PAULO
INSTITUTE OF MATHEMATICS AND STATISTICS
BACHELOR OF COMPUTER SCIENCE

Tractable Probabilistic Description Logic
Algorithms and Implementation

Andrew Ijano Lopes

FINAL ESSAY
MAC 499 — CAPSTONE PROJECT

Program: Computer Science

Advisor: Prof. Dr. Marcelo Finger

São Paulo
January 20th, 2021

Tractable Probabilistic Description Logic
Algorithms and Implementation

Andrew Ijano Lopes

This is the original version of the
capstone project report prepared by
the candidate Andrew Ijano Lopes, as
submitted to the Examining Committee.

Autorizo a reprodução e divulgação total ou parcial deste trabalho, por qualquer meio convencional ou eletrônico, para fins de estudo e pesquisa, desde que citada a fonte.

Resumo

Andrew Ijano Lopes. **Lógica de Descrição Probabilística Tratável: *Algoritmos e Implementação***. Monografia (Bacharelado). Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2020.

Elemento obrigatório, constituído de uma sequência de frases concisas e objetivas, em forma de texto. Deve apresentar os objetivos, métodos empregados, resultados e conclusões. O resumo deve ser redigido em parágrafo único, conter no máximo 500 palavras e ser seguido dos termos representativos do conteúdo do trabalho (palavras-chave). Deve ser precedido da referência do documento.

Palavras-chave: Lógicas de descrição. Palavra-chave2. Palavra-chave3.

Abstract

Andrew Ijano Lopes. **Tractable Probabilistic Description Logic: *Algorithms and Implementation***. Capstone Project Report (Bachelor). Institute of Mathematics and Statistics, University of São Paulo, São Paulo, 2020.

Elemento obrigatório, elaborado com as mesmas características do resumo em língua portuguesa. De acordo com o Regimento da Pós-Graduação da USP (Artigo 99), deve ser redigido em inglês para fins de divulgação. É uma boa ideia usar o sítio www.grammarly.com na preparação de textos em inglês.

Keywords: Description logics. Keyword2. Keyword3.

Lista de Abreviaturas

DL Description Logic

Lista de Símbolos

List of Figures

2.1	Graphical representation of the ontology in Example 2.2.	9
3.1	Annotation format for an uncertain axiom	13
3.2	Annotation format for a PBox restriction	14

List of Tables

2.1	Syntax and semantics of \mathcal{EL}^{++} without concrete domains	6
-----	--	---

List of Programs

Contents

1	Introduction	1
1.1	Objectives	2
1.2	Structure of this work	2
2	Background	3
2.1	Description logics	3
2.1.1	Building blocks of description logic ontologies	3
2.1.2	Description logic fragments and OWL	5
2.2	The description logic \mathcal{EL}^{++}	5
2.2.1	Syntax	5
2.2.2	Semantics	6
2.2.3	Normal form	6
2.3	Graphic \mathcal{EL} (\mathcal{GEL})	7
2.3.1	Graphical representation	8
2.3.2	SAT decision	9
2.4	MaxSAT for \mathcal{GEL}	9
2.5	Probabilistic \mathcal{GEL}	10
2.5.1	Linear algebraic view	10
3	Development	13
3.1	Input and output format	13
3.2	Knowledge base representation	14
3.3	PGEL-SAT solver	14
3.4	Linear solver	16
3.5	Column generation	17
3.5.1	Cost reduction	17
3.5.2	The algorithm	17
3.6	\mathcal{GEL} -MaxSAT solver	18

4	Experiments	19
5	Results	21
6	Related work	23
7	Conclusion and future work	25
References		27

Todo list

Need to rewrite, but the example is OK	1
mfinger: meus comentários em verde	1
Add initial description of the chapter	3
Maybe explain what is an ontology	3
Need to describe the graph completion rules	9
Question about this part: do we really need the graph completion to define the SAT decision? Isn't it true that a \mathcal{GEL} CBox \mathcal{C} is unsatisfiable iff $Init \rightsquigarrow \perp$ (using any type of edge) in a graph $G(\mathcal{C})$ without completion? I couldn't find any counterexample...	9
Numpy	13
Maybe explain how to execute the algorithm	13
Need to improve this	14
Need to say about M being large not the collection of possible columns	15
Explain the derivation of reduced cost	17

Chapter 1

Introduction

Need to rewrite, but the example is OK

mfinger: meus comentários em verde

Description logics are a family of formal knowledge representation languages, being of particular importance in providing a logical formalism for ontologies and the Semantic Web. Also, they are notable in biomedical informatics for assisting the codification of biomedical knowledge. Due to these uses, there is a great demand to find tractable (i.e., polynomial-time decidable) description logics.

One of them, the logic \mathcal{EL}^{++} , is one of the most expressive description logics in which the complexity of inferential reasoning is tractable (BAADER *et al.*, 2005). Even though it is expressive enough to deal with several practical applications, there was also a need to model situations in which a General Concept Inclusion Axiom is not always true, which has already been proposed in the literature (BOOLE, 1854).

Example 1.1

Consider a medical situation, in which a patient may have non-specific symptoms, such as high fever, cough, and headache. Also, COVID-19, a severe acute respiratory syndrome caused by the SARS-CoV-2 virus, is a disease that can account for those symptoms, but not all patients present all symptoms. Such an uncertain situation is suitable for probabilistic modeling.

In a certain hospital, a patient with a high fever has some probability of having COVID-19, but that probability is 20% larger if the patient has a cough too. On the other hand, COVID-19 is not very prevalent and is not observed in the hospital 90% of the time. If those probabilistic constraints are satisfiable, one can also ask the minimum and maximum probability that a hospital patient Mary, with fever and cough, is a suspect of suffering from COVID-19.

For classical propositional formulas, this problem, called *probabilistic satisfiability* (PSAT), has already been presented with tractable fragments (ANDERSEN and PRETOLANI, 2001). On the other hand, in description logics, most studies result in intractable reasoning;

moreover, by adding probabilistic reasoning capabilities to \mathcal{EL}^{++} , in order to model such situation, the complexity reaches NP-completeness (FINGER, 2019).

To solve this problem, probabilistic constraints can be applied to axioms and its probabilistic satisfaction can be seen in a linear algebraic view. Furthermore, it can be reduced to an optimization problem, which can be solved by an adaptation of the simplex method with column generation. (FINGER, 2019) Thus, it is possible to reduce the column generation problem to the *weighted partial maximum satisfiability*.

Then, recent studies show that it is necessary to focus on a fragment of \mathcal{EL}^{++} for obtain a tractable probabilistic reasoning. This fragment is called *Graphic \mathcal{EL}^{++} (\mathcal{GEL})* and it is defined as an \mathcal{EL}^{++} -fragment in which its set of axioms and *role inclusions* contains formulas in *normal form* and does not allow explicit conjunction axioms. Therefore, axioms can be seen as edges in a graph, as opposed to hyperedges in a hypergraph, which is the case of \mathcal{EL}^{++} . This allows the use of graph-based machinery to develop tractable algorithm for the *weighted partial Maximum SATisfatibility* for \mathcal{GEL} (Max \mathcal{GEL} -SAT) and, as a result, a tractable probabilistic description logic.

1.1 Objectives

- Investigate a potentially tractable fragment of \mathcal{EL}^{++} ;
- Study and implement tractable algorithms for the problem of *weighted partial Max \mathcal{GEL} -SAT*;
- Study and implement algorithms for the problem of *probabilistic satisfiability for \mathcal{GEL}* (PGEL-SAT), using the Max \mathcal{GEL} -SAT solver as a subroutine. Thus, it is expected to achieve a tractable algorithm for a probabilistic description logic.

1.2 Structure of this work

In this paper, we describe the implementation of these algorithms¹ and is organized as follows: Section ?? highlights related results in the literature. The basic definition of \mathcal{GEL} with its algorithms for MaxSAT and PSAT are described in Section ?? and followed by Section ??, which presents details about the implementation and its experimental evaluation.

¹Available at <https://github.com/AndrewIjano/pgel-sat>

Chapter 2

Background

In this chapter, we present the theoretical background of description logics,

Add initial description of the chapter

2.1 Description logics

Description logics (DLs) are used to represent knowledge, such as the semantic of words, people and their relations, and medical terms. These scenarios require precise specification and meaning so that different systems behave the same way. The first DL modeling languages appeared in the mid-1980s and have an important role in the context of the Semantic Web, an initiative to represent web content in a form that is more machine friendly (KRÖTZSCH *et al.*, 2012).

As their name suggests, DLs are logics; indeed, most of them are fragments of first-order logic. This relation with logics is what provides their precise specification, called *formal semantics*. Also, it equips their languages with a formal deduction to *infer* additional information, and the computation of these inferences is called *reasoning*. The performance of algorithms for reasoning strongly relies on the expressiveness of the DL: fast algorithms usually exist for lightweight logics. Then, there is not just a single DL because the balance between expressiveness and performance depends on the application. (KRÖTZSCH *et al.*, 2012)

2.1.1 Building blocks of description logic ontologies

A DL is composed of concepts, roles, and individual names. Concepts are sets of individuals, roles are binary relations between individuals and individual names are single individuals in the domain.

For example, an ontology modeling the situation in [Example 1.1](#) can use the concepts Patient, to represent the set of all patients in the hospital, and Symptom, to represent the set of all symptoms; roles such hasSymptom, to represent the binary relation between

Maybe explain what is an ontology

patients and symptoms; and individual names such as *mary* and *s1*, to represent the individuals Mary and Mary's symptoms.

Additionally, DLs allows us to describe more complex situations, creating new concepts and roles from the previously defined ones.

Some concept constructors provide boolean operations similar to that found in set theory and logic expressions. For example, if we want to describe the set of individuals that are both fever and cough, we could use the *conjunction* operator, as follows

$$\text{Fever} \sqcap \text{Cough}.$$

We can link concepts and roles using role restrictions. For example, to describe all individuals that are suspect of some disease that is COVID-19, we use the *existential restriction*

$$\exists \text{suspectOf.COVID-19}.$$

Also, to define concepts with only one individual we use *nominals* like $\{\text{mary}\}$.

More expressive logics can have other operations such as *disjunction* ($C \sqcup D$), *negation* ($\neg C$), *universal restriction* ($\forall r.C$) and *number restrictions* ($\leq n r.C$).

To capture knowledge about the world, DL ontologies also allow us to describe relations between concepts, roles, and individual names. For example, the fact that all fevers are symptoms is represented by the *concept inclusion*

$$\text{Fever} \sqsubseteq \text{Symptom};$$

the knowledge that someone that has a symptom which is caused by some disease is suspect of that disease can be expressed by the *role inclusion* with a *role composition*

$$\text{hasSymptom} \circ \text{hasCause} \sqsubseteq \text{suspectOf};$$

and the fact that Mary is a patient of the hospital and has symptoms is represented by the *assertions* $\text{Patient}(\text{mary})$ and $\text{hasSymptom}(\text{mary}, \text{s1})$.

After that, if we have a set of these relations, one could ask if there is a set of individuals, or instances, that satisfies these relations, which is called an *interpretation*. Interpretations can be understood as the assignment of meaning to logical terms in an ontology. Because a DL usually considers all the possible situations, property that is sometimes referred to as *open world assumption*, an ontology can have multiple satisfiable interpretations. The fewer restrictions it has, the more interpretations satisfy this ontology. The computational complexity to find the existence of these interpretations is one of the key aspects to choose different DL fragments.

These terms will be formally defined in the [section 2.2](#), in the case of the DL \mathcal{EL}^{++} .

2.1.2 Description logic fragments and OWL

There are many DL fragments. Each subset of features, like those described previously, can lead to different fragments of first-order logic. For example, the logic \mathcal{ALC} does not allow role inclusions and admits only \sqcap , \sqcup , \neg , \exists and \forall as concept constructors; their best reasoning algorithms, however, are worst-case exponential time. On the other hand, the \mathcal{EL} logic allows only \sqcap and \exists as concept constructors, and its reasoning algorithms are polynomial time.

To express DL ontologies, the World Wide Web Consortium (W3C) designed the OWL 2 Web Ontology Language (OWL 2) (HITZLER *et al.*, 2009). This declarative language is part of the W3C's Semantic Web technology stack and comes with various syntaxes, such as RDF/XML. Because of this use on the web, names in OWL are *international resource identifiers* (IRIs).

2.2 The description logic \mathcal{EL}^{++}

\mathcal{EL}^{++} is an extension of the DL \mathcal{EL} (BAADER *et al.*, 2005). It was created with large bio-health ontologies in mind, such as SNOMED-CT, the NCI thesaurus, and Galen, and became an official OWL 2 profile (HITZLER *et al.*, 2009). We concentrate on presenting \mathcal{EL}^{++} without concrete domains.

2.2.1 Syntax

In \mathcal{EL}^{++} , *concept descriptions* are defined inductively from a set N_C of *concept names*, a set N_R of *role names* and set N_I of *individual names* as follows:

- \top , \perp and concept names in N_C are concept descriptions;
- if C and D are concept descriptions, $C \sqcap D$ is a concept description;
- if C is a concept description and $r \in N_R$, $\exists r.C$ is a concept description;
- if $a \in N_I$, $\{a\}$ is a concept description.

To represent knowledge using concept descriptions, we need to define facts (axioms and role inclusions) and assertions.

An *axiom*, or a *general concept inclusion* (GCI), is an expression of the form $C \sqsubseteq D$, where C and D are concept inclusions. Also, we write $C \equiv D$ to represent the axioms $C \sqsubseteq D$ and $D \sqsubseteq C$. A *role inclusion* (RI) is an expression of the form $r_1 \circ \dots \circ r_k \sqsubseteq r$, where $r_1, \dots, r_k, r \in N_R$. The symbol “ \circ ” denotes composition of binary relations. A *constraint box* (CBox) is a finite set of GCIs and a finite set of RIs.

Similarly, a *concept assertion* is an expression of the form $C(a)$ and a *role assertion*, $r(a, b)$, where C is a concept description, $a, b \in N_I$ and $r \in N_R$. A finite set of concept assertions and role assertions is an *assertional box* (ABox).

Then, an \mathcal{EL}^{++} *knowledge base* \mathcal{K} (KB) is a pair (C, \mathcal{A}) , where C is a CBox and \mathcal{A} is an ABox.

2.2.2 Semantics

The semantics of \mathcal{EL}^{++} are given by *interpretations* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$. The *domain* $\Delta^{\mathcal{I}}$ is a non-empty set of individuals, and the *interpretation function* $\cdot^{\mathcal{I}}$ maps each concept name $A \in \mathbf{N}_C$ to a subset $A^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$, each role name r to a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, and each individual name a to an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. The extension of $\cdot^{\mathcal{I}}$ for an arbitrary concept description is inductively defined by the third column of Table 2.1.

	Name	Syntax	Semantics
	top	\top	Δ^I
	bottom	\perp	\emptyset
	nominal	$\{a\}$	$\{a^I\}$
	conjunction	$C \sqcap D$	$C^I \cap D^I$
existential restriction	$\exists r.C$	$\{x \in \Delta^I \mid \exists y \in \Delta^I : (x, y) \in r^I \wedge y \in C^I\}$	
	GCI	$C \sqsubseteq D$	$C^I \subseteq D^I$
	RI	$r_1 \circ \dots \circ r_k \sqsubseteq r$	$r_1^I \circ \dots \circ r_k^I \subseteq r^I$
concept assertion	$C(a)$	$a^I \in C^I$	
role assertion	$r(a, b)$	$(a^I, b^I) \in r^I$	

Table 2.1: Syntax and semantics of \mathcal{EL}^{++} without concrete domains

The interpretation \mathcal{I} satisfies:

- an axiom $C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ (represented as $\mathcal{I} \models C \sqsubseteq D$);
- a RI $r_1 \circ \dots \circ r_k \sqsubseteq r$ if $r_1^{\mathcal{I}} \circ \dots \circ r_k^{\mathcal{I}} \subseteq r^{\mathcal{I}}$ (represented as $\mathcal{I} \models r_1 \circ \dots \circ r_k \sqsubseteq r$);
- an assertion $C(a)$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$ (represented as $\mathcal{I} \models C(a)$);
- an assertion $r(a, b)$ if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$ (represented as $\mathcal{I} \models r(a, b)$).

Also, we say that \mathcal{I} is a *model* of:

- a CBox \mathcal{C} if it satisfies every axiom and RI in \mathcal{C} (represented as $\mathcal{I} \models \mathcal{C}$);
- an ABox \mathcal{A} if it satisfies every assertion in \mathcal{A} (represented as $\mathcal{I} \models \mathcal{A}$);

Then, an important problem in \mathcal{EL}^{++} is to determine its *consistency*, that is if \mathcal{A} and \mathcal{C} have a common model, which is in PTime (BAADER *et al.*, 2005).

2.2.3 Normal form

We can convert an \mathcal{EL}^{++} knowledge base into a normal form, in polynomial time, by introducing new concept and role names (BAADER *et al.*, 2005).

First, there is no need of explicit ABox, because $\mathcal{I} \models C(a) \iff \mathcal{I} \models \{a\} \sqsubseteq C$ and $\mathcal{I} \models r(a, b) \iff \{a\} \sqsubseteq \exists r.\{b\}$. In other words, a knowledge base can be represented by just a CBox, by transforming assertions in axioms.

In addition, given a CBox \mathcal{C} , consider the set $\text{BC}_{\mathcal{C}}$ of *basic concept descriptions*, which is the smallest set of concept descriptions that contains the top concept \top , all concept names used in \mathcal{C} and all concepts of the form $\{a\}$ used in \mathcal{C} .

Then, every axiom can be represented in the following normal form, where $C_1, C_2 \in \text{BC}_{\mathcal{C}}$, $D \in \text{BC}_{\mathcal{C}} \cup \{\perp\}$:

$$\begin{array}{ll} C_1 \sqsubseteq D & \text{(simple)} \\ C_1 \sqsubseteq \exists r.C_2 & \text{(existential-head)} \\ \exists r.C_1 \sqsubseteq D & \text{(existential-body)} \\ C_1 \sqcap C_2 \sqsubseteq D & \text{(conjunctive-body)} \end{array}$$

And every RI are of the form $r \sqsubseteq s$ or $r_1 \circ r_2 \sqsubseteq s$.

Example 2.1

Consider the following CBox \mathcal{C}_{exa} representing the situation in [Example 1.1](#). On the left, we have basic knowledge of diseases and, on the right, the specific knowledge about Mary. Note that, for simplicity, it is not in normal form.

$$\begin{array}{ll} \text{Fever} \sqsubseteq \text{Symptom} & \\ \text{Cough} \sqsubseteq \text{Symptom} & \\ \text{COVID-19} \sqsubseteq \text{Disease} & \{mary\} \sqsubseteq \text{Patient} \\ \text{Symptom} \sqsubseteq \exists \text{hasCause.Disease} & \{s1\} \sqsubseteq \text{Fever} \sqcap \text{Cough} \\ \text{Patient} \sqsubseteq \exists \text{hasSymptom.Symptom} & \{mary\} \sqsubseteq \exists \text{hasSymptom}.\{s1\} \\ \text{hasSymptom} \circ \text{hasCause} \sqsubseteq \text{suspectOf} & \end{array}$$

Because CBoxes can only represent facts, there is no way to describe uncertain knowledge. Even though, in cases when which of them are true, we could define three axioms

$$\begin{array}{l} Ax_1 := \text{Fever} \sqsubseteq \exists \text{hasCause.COVID-19}, \text{ when fever is actually caused by COVID-19;} \\ Ax_2 := \text{Fever} \sqcap \text{Cough} \sqsubseteq \exists \text{hasCause.COVID-19}, \text{ when both fever and cough are caused by COVID-19;} \\ Ax_3 := \text{COVID-19} \sqsubseteq \perp, \text{ when there are no presence of COVID-19 in the hospital.} \end{array}$$

In the following sections, it will be presented how to add these axioms in a KB with probabilistic properties.

We want to model uncertain information using DLs. However, it has been proved that, by adding probabilistic reasoning capabilities to \mathcal{EL}^{++} , the complexity reaches NP-completeness ([FINGER, 2019](#)). Then, it is necessary to reduce the expressiveness of this language.

2.3 Graphic \mathcal{EL} (\mathcal{GEL})

Graphic \mathcal{EL} (\mathcal{GEL}) is a fragment of \mathcal{EL}^{++} in which every axiom and RI are in normal form and only simple and existential-head axioms are allowed ([FINGER, n.d.](#)). The semantics

are the same as that of \mathcal{EL}^{++} .

Example 2.2

Since there are conjunctive-body axioms in the CBox in [Example 2.1](#), we need to modify this knowledge in order to represent it in \mathcal{GEL} . First, we substitute every concept description $\text{Fever} \sqcap \text{Cough}$ by a new basic concept FeverAndCough . After that, we add axioms $\text{FeverAndCough} \sqsubseteq \text{Fever}$ and $\text{FeverAndCough} \sqsubseteq \text{Cough}$ to CBox.

The name of this fragment comes from the fact that each GCI can be represented as arrows in a graph where nodes are basic concepts. This representation is useful for the development of algorithms and it is used to define its SAT decision.

2.3.1 Graphical representation

Consider a \mathcal{GEL} -CBox C with n_R roles, its graphical representation is a edge-labeled graph $G(C) := (N, E, \ell)$, where N is a set of nodes, $E \subseteq N^2$ is a set of directed edges and $\ell : E \rightarrow \{0, 1, \dots, n_R\}$ is a labeling function.

In addition, **FINGER** (n.d.) defines the following notation. The set $E_i \subseteq E$ is the set of all edges e such that $\ell(e) = i$. We write $X_1 \rightarrow_i X_2$ if there is an edge $(X_1, X_2) \in E_i$, and $X_1 \not\rightarrow_i X_2$ if $(X_1, X_2) \notin E_i$. The expression $X \rightarrow_i^* Y$ represents the reflexive transitive closure of \rightarrow_i , which is the existence of a path in the graph of size ≥ 0 , starting in X , ending in Y , and only going through edges in E_i , for $0 \leq i \leq n_R$. Finally, $X \rightsquigarrow Y$ represents a path from X to Y using any type of edge.

Then, the graph $G(C) = (N, E, \ell)$ can be constructed from a CBox C with the following steps:

1. for each concept $\exists r.C$ in a existential-body axiom of the form $\exists r.C \sqsubseteq D$ create a *virtual concept* “ $\exists r.C$ ”. The set of all virtual concepts in C is called VC_C ;
2. the set N of nodes is obtained from the basic concepts of C , an initial-node symbol $Init$, the bottom concept \perp and the set of virtual concepts of C as follows:

$$N := \{Init, \perp\} \cup BC_C \cup VC_C;$$

3. if $C \sqsubseteq D \in C$ then $C \rightarrow_0 D$;
4. if $C \sqsubseteq \exists r_i.D$ then $C \rightarrow_i D$;
5. $Init \rightarrow_0 \top$;¹
6. for every node of the form $\{a_i\} \in N$, $Init \rightarrow_0 \{a_i\}$.

Example 2.3

Consider the CBox in [Example 2.2](#) and the uncertain information in [Example 2.1](#). Its graphical representation is displayed in the [Figure 2.1](#). The \rightarrow_0 -edges are represented by continuous black arrows, the \rightarrow_i -edges, $i \geq 1$, are represented by blue labeled arrows and red dotted arrows indicate that source is uncertain information.

¹Talk about sat

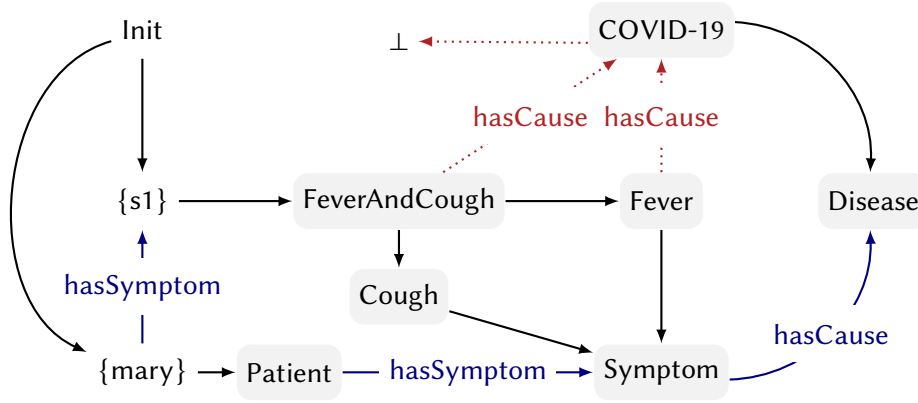


Figure 2.1: Graphical representation of the ontology in *Example 2.2*.

2.3.2 SAT decision

Need to describe the graph completion rules

Question about this part: do we really need the graph completion to define the SAT decision? Isn't it true that a \mathcal{GEL} CBox C is **unsatisfiable** iff $Init \rightsquigarrow \perp$ (using any type of edge) in a graph $G(C)$ **without** completion? I couldn't find any counterexample...

After the graph completion, the satisfiability (SAT) decision of a \mathcal{GEL} CBox can be reduced to a path search in a graph, that is, a \mathcal{GEL} CBox is unsatisfiable iff $Init \rightarrow_0^* \perp$.

2.4 MaxSAT for \mathcal{GEL}

Before focusing on the probabilistic extension for \mathcal{GEL} , we need to define its MaxSAT problem, which will compose further the probabilistic reasoner.

The *weighted partial maximum satisfiability problem for \mathcal{GEL}* (\mathcal{GEL} -MaxSAT) can be defined as follows: given a potentially inconsistent weighted CBox C , we want to find the maximal satisfiable subset of axioms; since it is partial, some axioms must be present in this subset. Usually, partiality can be modeled assigning infinite weights to the axioms that must not be excluded.

A *weighted* CBox is a pair $\langle C, w \rangle$ where C is a CBox and $w : C \rightarrow \mathbb{Q} \cup \{\infty\}$ is a weight function, which maps axioms in C to weights. The infinite weight is used to represent axioms that must not be excluded in the maximal satisfiable subset. Also, it is defined that RIs of the form $r \sqsubseteq s$ and $r_1 \circ r_2 \sqsubseteq s$ always have infinite weight.

Then, given a weighted CBox $\langle C, w \rangle$, a solution for the weighted partial \mathcal{GEL} -MaxSAT problem is a set $C_{max} \subseteq C$ such that:

- C_{max} is satisfiable; and
- $C_{max} \models C \sqsubseteq D$ if $w(C \sqsubseteq D) = \infty$; and
- the sum of finite weights in C_{max} is maximal.

2.5 Probabilistic \mathcal{GEL}

Probability in \mathcal{GEL} is constructed from a *probability function* P (FINGER, n.d.). Consider a finite number of interpretation, $\mathcal{I}_1, \dots, \mathcal{I}_m$, we define the probability function $P : \{\mathcal{I}_1, \dots, \mathcal{I}_m\} \rightarrow \mathbb{Q}$, such that $P(\mathcal{I}_i) \geq 0$ and $\sum_{i=1}^m P(\mathcal{I}_i) = 1$. We can also define the probability of an axiom $C \sqsubseteq D$ as follows

$$P(C \sqsubseteq D) = \sum_{\mathcal{I}_i \models C \sqsubseteq D} P(\mathcal{I}_i).$$

A *probabilistic knowledge base* is a pair $\langle C, \mathcal{P} \rangle$, where C is a CBox and \mathcal{P} is a PBox. A PBox is a set of k linear constraints over n axioms, of the form

$$\sum_{j=1}^n a_{ij} \cdot P(C_j \sqsubseteq D_j) \leq b_i; \quad 1 \leq i \leq k. \quad (2.1)$$

We can define the *satisfiability problem* for this probabilistic KB (PGEL-SAT) as deciding if it is consistent or not. If it is consistent, the solution is a set of interpretations $\{\mathcal{I}_1, \dots, \mathcal{I}_m\}$ and a probability function $P : \{\mathcal{I}_1, \dots, \mathcal{I}_m\} \rightarrow \mathbb{Q}^+$ such that $\sum_{i=1}^m P(\mathcal{I}_i) = 1$, $P(C \sqsubseteq D) = 1$ for $C \sqsubseteq D \in C$ (axioms in CBox are certain) and P verifies all linear constraints in \mathcal{P} .

Example 2.4

Now we can model the uncertain situation stated in Example 1.1 using the probability knowledge base $\langle C_{exa}, \mathcal{P}_{exa} \rangle$, where C_{exa} is the CBox from Example 2.2 and \mathcal{P}_{exa} is given by

$$\mathcal{P}_{exa} := \left\{ \begin{array}{l} P(Ax_2) - P(Ax_1) = 0.2, \\ P(Ax_3) = 0.9 \end{array} \right\}.$$

Then, we need a polynomial algorithm to find if this probabilistic KB is consistent.

2.5.1 Linear algebraic view

The PGEL-SAT problem was also defined by FINGER (n.d.) in a linear algebraic view, which is useful to develop its polynomial reasoning algorithm. It was shown that a probabilistic KB $\langle C, \mathcal{P} \rangle$ is satisfiable iff the linear equation $C \cdot x = d$ has a solution $x \geq 0$, where

$$C := \begin{bmatrix} -I_n & M_{n \times m} \\ A_{k \times n} & 0_{k \times m} \\ 0'_n & 1'_m \end{bmatrix} \quad x := \begin{bmatrix} p_n \\ \pi_m \end{bmatrix} \quad d := \begin{bmatrix} 0_n \\ b_k \\ 1 \end{bmatrix} \quad (2.2)$$

and

- $A_{k \times n}$ is a $k \times n$ matrix whose elements a_{ij} are given by Equation 2.1;
- b_k is a k vector whose elements b_i are also given by Equation 2.1;

- $M_{n \times m}$ is a $n \times m$ matrix given by the following steps:

Consider an interpretation \mathcal{I} model of \mathcal{C} , its corresponding vector in \mathcal{P} is a $\{0, 1\}$ -vector y such that $y_i = 1$ iff $\mathcal{I} \models C_i \sqsubseteq D_i$, for $1 \leq i \leq n$.

Then, given a set of interpretations $\mathcal{I}_1, \dots, \mathcal{I}_m$, we define $M_{n \times m}$ a matrix whose column M^j is \mathcal{I}_j 's corresponding vector in \mathcal{P} ;

- I_n is the n -dimensional identity matrix;
- 0_n is a column 0-vector of size n (similarly for 1_n);
- $0'_n$ is the previous vector's transpose;
- $0_{k \times m}$ is a 0-matrix of shape $k \times m$;
- p_n is a vector of size n which corresponds to the probability of axioms occurring in [Equation 2.1](#);
- π_m is a vector of size m which corresponds to the probability distribution over interpretations $\mathcal{I}_1, \dots, \mathcal{I}_m$.

Thus, we can use techniques for solving linear equations to find a tractable algorithm for PGEL-SAT.

Chapter 3

Development

In this section, we describe the development of a tractable algorithm for PGEL-SAT¹, described by FINGER (n.d.). It was implemented using Python programming language (VAN ROSSUM and DRAKE, 2009).

Numpy

3.1 Input and output format

The algorithm accepts as input a $\mathcal{P}\mathcal{G}\mathcal{E}\mathcal{L}$ KB encoded in an OWL 2 ontology. Both certain and uncertain knowledge must be a $\mathcal{G}\mathcal{E}\mathcal{L}$ -CBox in the normal form, with the additional support of equivalence axioms. Due to limitations in the OWL parser used, which will be detailed further, RDF/XML, OWL/XML and NTriples are the only file formats supported.

In addition, uncertain axioms must have an annotation (`rdfs:comment`) with its unique numerical index. That is, given an uncertain axiom Ax_i , its annotation must be of the following form in Figure 3.1.

`#!pbox-id i`

Figure 3.1: Annotation format for an uncertain axiom

PBox restrictions are represented with annotations in the \top concept (`owl:Thing`). That is, given a restriction of the form $a_0P(Ax_0) + a_1P(Ax_1) + \dots + a_{n-1}P(Ax_{n-1}) = b$, its annotation must be of the form in Figure 3.2. Also, inequalities \leq and \geq can be represented, respectively, with the symbols `<=` and `>=`.

The output of the algorithm is `True` if the given KB has a solution, and `False` if not.

Maybe explain how to execute the algorithm

¹Available at <https://github.com/AndrewIjano/pgel-sat>.

```

#!pbox-restriction
0  a0
1  a1
...
n - 1  an-1
==
b

```

Figure 3.2: Annotation format for a PBox restriction

3.2 Knowledge base representation

To read the ontology in OWL 2, it was used the Python module *Owlready2* (LAMY, 2017).

The probabilistic KB is represented as the edge-labeled graph in subsection 2.3.1 with three matrices for the inequalities in Equation 2.1.

Need to improve this

The probabilistic KB representation allows us to develop a PGEL-SAT solver, using this data structure as input.

3.3 PGEL-SAT solver

In order to understand the PGEL-SAT solver implemented, we need to find a solution for the linear system stated in subsection 2.5.1.

First, consider the matrix C in Equation 2.1, it has p -columns and π -columns, referring to which part of x they multiply. We say that a p -column pc_i is *well-formed* if it is of the form in Equation 2.1, starting with the i -th column of $-I_n$, followed by the i -th column of A and with one 0 at the end. Also, we say that a π -column is *well-formed* if it starts with a corresponding vector of an interpretation model of C , followed by k zeros and with one 1 at the end. A column that is not well-formed is *ill-formed*.

Now, consider that the matrix C may have ill-formed columns. We define a binary *cost vector* c such that each element $c_i = 1$ iff column C^i is ill-formed; otherwise $c_i = 0$. Then, the linear system stated in subsection 2.5.1 has a solution iff the following minimization problem has minimum 0.

$$\begin{aligned}
 &\text{minimize} && c' \cdot x \\
 &\text{subject to} && C \cdot x = d \\
 &&& x \geq 0
 \end{aligned} \tag{3.1}$$

This problem can be solved by a linear algebraic solver but we need to find beforehand C and d that reaches the minimum 0. Since matrix A and vector b are generated from

PBox, we need to choose a set of interpretations that provides the solution. However, we potentially have an exponential collection of possible interpretations, and looking over all of them would make the algorithm untractable.

Need to say about M being large not the collection of possible columns

Even though, from Carathéodory's Theorem (ECKHOFF, 1993), FINGER (n.d.) states that if constraints in Equation 2.1 are solvable then there exists a solution where x has at most $n + k + 1$ values such that $x_j > 0$. This allows us to avoid generating an exponential number of columns of C by using a linear algebraic technique called *column generation* (GILMORE and GOMORY, 1961; GILMORE and GOMORY, 1963). With this technique, we can create an algorithm PGEL-SAT-SOLVER which generates well-formed π -columns of C on the fly by solving an auxiliary problem and uses a linear solver in each iteration to verify if the solution for Equation 3.1 has reached minimum 0. This solution leads to the algorithm implemented, which is shown in Algorithm 1.

Algorithm 1 The PGEL-SAT solver algorithm

```

1: function PGEL-SAT-SOLVER( $\langle C, \mathcal{P} \rangle$ )
2:    $c \leftarrow \text{INITIALIZE-C}(\mathcal{P})$ 
3:    $C \leftarrow \text{INITIALIZE-C}(\mathcal{P})$ 
4:    $d \leftarrow \text{INITIALIZE-D}(\mathcal{P})$ 

5:    $lp \leftarrow \text{LINPROG}(c, C, d)$ 

6:   while  $lp.minCost \neq 0$  do
7:      $result \leftarrow \text{GENERATECOLUMN}(\langle C, \mathcal{P} \rangle, lp)$ 
8:     if  $result.isSuccess = \text{False}$  then
9:       return  $\text{False}$   $\triangleright \langle C, \mathcal{P} \rangle$  is unsatisfiable
10:    end if

11:     $y \leftarrow result.column$   $\triangleright$  well-formed  $\pi$ -column
12:     $C \leftarrow (C | y)$ 
13:     $c \leftarrow (c | 0)$ 
14:     $lp \leftarrow \text{LINPROG}(c, C, d)$ 
15:  end while

16:  return  $(\text{True}, lp)$   $\triangleright \langle C, \mathcal{P} \rangle$  is satisfiable
17: end function

```

The algorithm PGEL-SAT-SOLVER in Algorithm 1 receives a potentially unsatisfiable probabilistic knowledge base and returns *True* if it is satisfiable, and *False*, otherwise.

It starts with matrices c , C and d . Because the algorithm needs to start with a cost vector with non-zero elements, the C matrix is initialized with $n + k + 1$ ill-formed columns, represented by the identity matrix I_{n+k+1} , and well-formed p -columns as shown in Equation 2.2; therefore, the cost vector c is initialized with $n + k + 1$ 1-elements plus n 0-elements. The d vector is unaltered during the execution and is given by Equation 2.2. These initializations are shown in Algorithm 2.

In addition, the algorithm uses a program LINPROG. It receives matrices c , C and d and

Algorithm 2 Matrices initializations

```

1: function INITIALIZE-C( $\mathcal{P}$ )
2:   return ( $1'_{n+k+1} \mid 0'_n$ )'
3: end function

4: function INITIALIZE-C( $\mathcal{P}$ )
5:   Get  $A$  from PBox  $\mathcal{P}$  restrictions
6:    $PC \leftarrow \begin{pmatrix} -I_n \\ A \\ 1'_n \end{pmatrix}$   $\triangleright$   $p$ -columns
7:   return ( $I_{n+k+1} \mid PC$ )
8: end function

9: function INITIALIZE-D( $\mathcal{P}$ )
10:  Get  $b$  from PBox  $\mathcal{P}$  restrictions
11:  return ( $0'_n \mid b' \mid 1$ )'
12: end function

```

returns the solution for the optimization problem in [Equation 3.1](#). Its output lp needs to contain the minimal cost ($lp.minCost$), the primal solution ($lp.primal$) and dual solution ($lp.dual$).

Then, the algorithm executes the column generation process while the minimal cost is not zero. In each iteration, a well-formed π -column y is generated by an auxiliary program GENERATECOLUMN. After that, the column is appended to C , with its respective 0 cost in c , and a new minimal cost is calculated. If no more columns can be generated with a minimal cost higher than 0, represented by $result.isSuccess = False$, the KB is unsatisfiable.

3.4 Linear solver

The program LINPROG was implemented using the open-source GNU linear program kit ([MAKHORIN, 2001](#)) and the Python wrapper for this library, the *swiglpk* package ([SONNENSCHEIN, n.d.](#)). The library was chosen because it is open-source, has tractable algorithms for large-scale problems, and provides a solution that contains the required elements described in [section 3.3](#).

There are several methods for solving a linear optimization problem. The *simplex method* is known to be very efficient in practice ([BORGWARDT, 2012](#)) although its worst time complexity is untractable ([KLEE and MINTY, 1972](#)). On the other hand, the modern *interior point method* has a proven polynomial-time complexity and is especially ideal for very sparse problems ([BOYD et al., 2004](#)) but may not be as efficient as the simplex method in all cases.

Because of the theoretical tractability, the linear solver LINPROG implemented uses the interior point method.

3.5 Column generation

The program `GENERATECOLUMN` is responsible for generating a well-formed π -column for C that reduces $c'x$ until it is minimal; in particular, when the KB is satisfiable, this value is equal to 0. To find such a column, we need to analyze the optimization problem we are solving.

3.5.1 Cost reduction

Suppose that the matrix C is composed of I_{n+k+1} and all well-formed p -columns PC , which are result of `INITIALIZE-C`, concatenated with all possible (ill and well formed) π -columns $y^{(i)}$, as illustrated below.

$$C := (I_{n+k+1} | PC | y^{(1)} | y^{(2)} \dots)$$

This matrix can be rearranged in the form $C = (B | N)$, where B is a linear independent square matrix called *basis matrix*. This allows obtaining a *basic feasible solution* $x = (x'_B | x'_N)'$ where $x_N = 0$ and $x_B = B^{-1}d$. In particular, if $B = I_{n+k+1}$ we have $x_B = d$.

However, this solution may, and probably will, not be optimal. Then, we need to find some column C^j in N and increase x_j by some positive value θ . For that, we create a vector s such that $s_j = 1$, $s_i = 0$ for every non basic index $i \neq j$ and its basic part $s_B = (s_{B(1)}, s_{B(2)}, \dots, s_{B(m)})$.

Explain the derivation of reduced cost

From that, we define that the *reduced cost* r_y of a column y to be added to the solution is

$$r_y = c_y - w \cdot y.$$

Because y is well-formed, we have $c_y = 0$. Then, to reduce the cost of the objective function, we need $r_y \leq 0$ and, therefore, the new column must satisfy

$$w \cdot y \leq 0.$$

3.5.2 The algorithm

- just the columns of M must be generated
- this is the corresponding vector of a interpretation of axioms in P . Similar to truth values from propositional logic.
- to maximize the cost reduction, we want to prioritize interpretations that model axioms associated with higher values in w
- this calculation can be done by solving the gel max sat problem where only axioms from $pbox$ have finite weight given by w .
- this leads to the algorithm in [Algorithm 4](#)

Algorithm 3 The algorithm of column generation

```

1: function GENERATECOLUMN( $\langle C, \mathcal{P} \rangle, lp$ )
2:    $w \leftarrow lp.dual$ 
3:    $result \leftarrow \mathcal{GEL}\text{-MAX-SAT-SOLVER}(\langle C, \mathcal{P} \rangle, w)$ 
4:   if  $result.isSuccess = \text{False}$  then
5:     return False
6:   end if

7:    $y \leftarrow \text{EXTRACTCOLUMN}(result)$ 
8:   if  $w \cdot y < 0$  then
9:     return False
10:  end if

11:  return (True,  $y$ )
12: end function

13: function EXTRACTCOLUMN( $result$ )
14:  Define  $n$ -vector  $m_n$  such that  $m_i = 1$  iff we have  $Ax_i$  in  $result$ 
15:   $y \leftarrow (m'_n \mid 0'_k \mid 1)'$ 
16:  return  $y$ 
17: end function

```

3.6 \mathcal{GEL} -MaxSAT solver

- explain the generation of the weighted graph G_w
- explain the theory of the solution
- pseudocode of the algorithm
- explain the min cut algorithm

Algorithm 4 The \mathcal{GEL} -MaxSAT solver algorithm

```

1: function  $\mathcal{GEL}\text{-MAX-SAT}(\langle C, \mathcal{P} \rangle, w)$ 
2:  Compute  $G_w(C, \mathcal{P}) \leftarrow (N_w, E_w, w)$ 
3:   $CS \leftarrow \text{MINCUT}(G_w(C, \mathcal{P}))$ 
4:  if  $w(CS) = \infty$  then
5:    return False
6:  end if

7:  return (True,  $CS.indexes$ )
8: end function

```

Chapter 4

Experiments

- tests to measure time complexity
- experiments for different algorithms

Chapter 5

Results

Chapter 6

Related work

The problem of probabilistic reasoning and extensions in logics to deal with uncertainty have been studied for several decades. The first known proposal of PSAT, for propositional formulas, is attributed to [BOOLE \(1854\)](#) and it has already been shown to be NP-Complete ([GEORGAKOPOULOS *et al.*, 1988](#)).

In the relational domain, the literature contain several logics with probabilistic reasoning capabilities although they have led to intractable decision problems. Some of them extend the already intractable \mathcal{ALC} , with probabilistic constraints over concepts ([HEINSOHN, 1994](#); [LUKASIEWICZ, 2008](#); [GUTIÉRREZ-BASULTO *et al.*, 2011](#)). For the expressive and lightweight \mathcal{EL} -family, some extensions such as [GUTIÉRREZ-BASULTO *et al.* \(2017\)](#) and [CEYLAN and PEÑALOZA \(2017\)](#) have led to ExpTime-hard or PP-complete probabilistic reasoning; furthermore, NP-completeness can be achieved with probability capabilities over axioms ([FINGER, 2019](#)).

On the other hand, many results implies that the research on Max-SAT has a impact on the solutions of PSAT problems ([ANDERSEN and PRETOLANI, 2001](#)). Also, there was already proposed a MaxSAT-solver for a propositional fragment of horn logic by a max-flow/min-cut formulation ([JAUARD and SIMEONE, 1987](#)). Thus, it is expected to ask if one could also take such results to a relational domain.

Chapter 7

Conclusion and future work

References

- [ANDERSEN and PRETOLANI 2001] Kim Allan ANDERSEN and Daniele PRETOLANI. “Easy cases of probabilistic satisfiability”. In: *Annals of Mathematics and Artificial Intelligence* 33.1 (2001), pp. 69–91 (cit. on pp. 1, 23).
- [BAADER *et al.* 2005] Franz BAADER, Sebastian BRANDT, and Carsten LUTZ. “Pushing the EL Envelope”. In: *Proceedings of the 19th International Joint Conference on Artificial Intelligence*. IJCAI’05. 2005, pp. 364–369 (cit. on pp. 1, 5, 6).
- [BOOLE 1854] G. BOOLE. *An Investigation of the Laws of Thought: On which are Founded the Mathematical Theories of Logic and Probabilities*. Collected logical works. Walton and Maberly, 1854. URL: <https://books.google.com.br/books?id=DqwAAAAcAAJ> (cit. on pp. 1, 23).
- [BORGWARDT 2012] Karl Heinz BORGWARDT. *The Simplex Method: a probabilistic analysis*. Vol. 1. Springer Science & Business Media, 2012 (cit. on p. 16).
- [BOYD *et al.* 2004] Stephen BOYD, Stephen P BOYD, and Lieven VANDENBERGHE. *Convex optimization*. Cambridge university press, 2004 (cit. on p. 16).
- [CEYLAN and PEÑALOZA 2017] Ismail Ilkan CEYLAN and Rafael PEÑALOZA. “The Bayesian Ontology Language BEL”. In: *Journal of Automated Reasoning* 58.1 (2017), pp. 67–95 (cit. on p. 23).
- [ECKHOFF 1993] Jürgen ECKHOFF. “Helly, Radon, and Carathéodory type theorems”. In: *Handbook of convex geometry*. Elsevier, 1993, pp. 389–448 (cit. on p. 15).
- [FINGER 2019] Marcelo FINGER. “Extending EL++ with Linear Constraints on the Probability of Axioms”. In: *Description Logic, Theory Combination, and All That. Essays Dedicated to Franz Baader on the Occasion of His 60th Birthday*. Ed. by Carsten LUTZ, Uli SATTler, Cesare TINELLI, Anni-Yasmin TURHAN, and Frank WOLTER. Vol. LLNCS 11560. Theoretical Computer Science and General Issues. Springer International Publishing, 2019. ISBN: 978-3-030-22101-0. DOI: [10.1007/978-3-030-22102-7](https://doi.org/10.1007/978-3-030-22102-7) (cit. on pp. 2, 7, 23).
- [FINGER n.d.] Marcelo FINGER. “Tractable Max-SAT in Graphic Description Logics”. In preparation (cit. on pp. 7, 8, 10, 13, 15).

- [GEORGAKOPOULOS *et al.* 1988] George GEORGAKOPOULOS, Dimitris KAVVADIAS, and Christos H PAPADIMITRIOU. “Probabilistic satisfiability”. In: *Journal of complexity* 4.1 (1988), pp. 1–11 (cit. on p. 23).
- [GILMORE and GOMORY 1961] Paul C GILMORE and Ralph E GOMORY. “A linear programming approach to the cutting-stock problem”. In: *Operations research* 9.6 (1961), pp. 849–859 (cit. on p. 15).
- [GILMORE and GOMORY 1963] Paul C GILMORE and Ralph E GOMORY. “A linear programming approach to the cutting stock problem—Part II”. In: *Operations research* 11.6 (1963), pp. 863–888 (cit. on p. 15).
- [GUTIÉRREZ-BASULTO *et al.* 2011] Víctor GUTIÉRREZ-BASULTO, Jean Christoph JUNG, Carsten LUTZ, and Lutz SCHRÖDER. “A Closer Look at the Probabilistic Description Logic Prob-EL”. In: *Proc. 25th Conference on Artificial Intelligence (AAAI-11)*. Ed. by Wolfram BURGARD and Dan ROTH. AAAI Press, 2011, pp. 197–202 (cit. on p. 23).
- [GUTIÉRREZ-BASULTO *et al.* 2017] Víctor GUTIÉRREZ-BASULTO, Jean Christoph JUNG, Carsten LUTZ, and Lutz SCHRÖDER. “Probabilistic description logics for subjective uncertainty”. In: *Journal of Artificial Intelligence Research* 58 (2017), pp. 1–66 (cit. on p. 23).
- [HEINSOHN 1994] Jochen HEINSOHN. “Probabilistic description logics”. In: *Uncertainty Proceedings 1994*. Elsevier, 1994, pp. 311–318 (cit. on p. 23).
- [HITZLER *et al.* 2009] Pascal HITZLER, Markus KRÖTZSCH, Bijan PARSIA, Peter F PATEL-SCHNEIDER, Sebastian RUDOLPH, *et al.* “OWL 2 web ontology language primer”. In: *W3C recommendation* 27.1 (2009), p. 123 (cit. on p. 5).
- [JAUMARD and SIMEONE 1987] Brigitte JAUMARD and Bruno SIMEONE. “On the complexity of the maximum satisfiability problem for Horn formulas”. In: *Information Processing Letters* 26.1 (1987), pp. 1–4 (cit. on p. 23).
- [KLEE and MINTY 1972] Victor KLEE and George J MINTY. “How good is the simplex algorithm”. In: *Inequalities* 3.3 (1972), pp. 159–175 (cit. on p. 16).
- [KRÖTZSCH *et al.* 2012] Markus KRÖTZSCH, Frantisek SIMANCIK, and Ian HORROCKS. “A description logic primer”. In: *arXiv preprint arXiv:1201.4089* (2012) (cit. on p. 3).
- [LAMY 2017] Jean-Baptiste LAMY. “Owlready: Ontology-oriented programming in Python with automatic classification and high level constructs for biomedical ontologies”. In: *Artificial intelligence in medicine* 80 (2017), pp. 11–28 (cit. on p. 14).
- [LUKASIEWICZ 2008] Thomas LUKASIEWICZ. “Expressive probabilistic description logics”. In: *Artificial Intelligence* 172.6-7 (2008), pp. 852–883 (cit. on p. 23).

REFERENCES

- [MAKHORIN 2001] Andrew MAKHORIN. “GNU linear programming kit”. In: *Moscow Aviation Institute, Moscow, Russia* 38 (2001) (cit. on p. 16).
- [SONNENSCHN n.d.] Nikolaus SONNENSCHN. *swiglpk - Simple swig bindings for the GNU Linear Programming Kit*. URL: <https://pypi.org/project/swiglpk/> (cit. on p. 16).
- [VAN ROSSUM and DRAKE 2009] Guido VAN ROSSUM and Fred L. DRAKE. *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009. ISBN: 1441412697 (cit. on p. 13).