PRACTICAL REASONING IN PROBABILISTIC DESCRIPTION LOGIC

A thesis submitted to the University of Manchester
for the degree of Doctor of Philosophy
in the Faculty of Engineering and Physical Sciences

2011

By
Pavel Klinov
School of Computer Science

Contents

List of Tables

List of Figures

Abstract

Description Logics (DLs) form a family of languages which correspond to decidable fragments of First-Order Logic (FOL). They have been overwhelmingly successful for constructing ontologies—conceptual structures describing domain knowledge. Ontologies proved to be valuable in a range of areas, most notably, bioinformatics, chemistry, Health Care and Life Sciences, and the Semantic Web.

One limitation of DLs, as fragments of FOL, is their restricted ability to cope with various forms of uncertainty. For example, medical knowledge often includes statistical relationships, e.g., findings or results of clinical trials. Currently it is maintained separately, e.g., in Bayesian networks or statistical models. This often hinders knowledge integration and reuse, leads to duplication and, consequently, inconsistencies.

One answer to this issue is *probabilistic logics* which allow for smooth integration of classical, i.e., expressible in standard FOL or its sub-languages, and uncertain knowledge. However, probabilistic logics have long been considered impractical because of discouraging computational properties. Those are mostly due to the lack of simplifying assumptions, e.g., independence assumptions which are central to Bayesian networks.

In this thesis we demonstrate that deductive reasoning in a particular probabilistic DL, called P-$\mathcal{SROIQ}$, can be computationally practical. We present a range of novel algorithms, in particular, the probabilistic satisfiability procedure (PSAT) which is, to our knowledge, the first scalable PSAT algorithm for a non-propositional probabilistic logic. We perform an extensive performance and scalability evaluation on different synthetic and natural data sets to justify practicality.

In addition, we study theoretical properties of P-$\mathcal{SROIQ}$ by formally translating it into a fragment of first-order logic of probability. That allows us to gain a better insight into certain important limitations of P-$\mathcal{SROIQ}$. Finally, we investigate its applicability from the practical perspective, for instance, use it to extract all inconsistencies from a real rule-based medical expert system.

We believe the thesis will be of interest to developers of probabilistic reasoners. Some of the algorithms, e.g., PSAT, could also be valuable to the Operations Research community since they are heavily based on mathematical programming. Finally, the theoretical analysis could be helpful for designers of future probabilistic logics.

Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright

Acknowledgments

I would like to thank my supervisors Bijan Parsia and Uli Sattler, who have been truly wonderful during the entire course of my PhD studies at Manchester. Special thanks go to David Picado-Muiño whose involvement was instrumental for our work on CADIAG-2. Finally, I am deeply indebted to both my examiners, Konstantin Korovin and Ralf Möller, for their extremely thorough reading of this thesis and the great discussion during the defense (which I enjoyed a lot more than I could have ever anticipated).

Chapter 1

Introduction

This thesis is primarily concerned with investigating *practical* optimization techniques and application scenarios for deductive reasoning in probabilistic Description Logics. Both probability theory and formal logic are of paramount importance to AI and have received tremendous research attention. It is no surprise that plenty of ways to combine the two have been proposed. Unfortunately, combined formalisms, especially those which present the representational power of logic, tend to be computationally much harder than each of the components often leading to the claim that probabilistic logics are inherently impractical. A major part of our goal is to show that this claim is not universally true.

This chapter aims to lay out relevant aspects of both Description Logics (DLs) and uncertainty management in knowledge representation (KR) systems. We first give a brief informal overview of DLs, in particular, the long way they went from early KR systems to modern ontology languages and highly optimized reasoners. Then it explains the ubiquity of uncertainty, describes its numerous kinds and flavors paying a particular attention to the important distinction between probability and fuzzy membership. Our position is that uncertainty is an intrinsic feature of background knowledge and thus it is natural to capture it in ontologies rather than in separate models, e.g., Bayesian networks. The success of such an approach is, of course, contingent upon acceptable performance of available reasoning tools.

1.1 Description Logic and Ontologies

Description Logics [9] is a family of languages which correspond to (decidable) fragments of first-order logic (FOL).[1] DLs offer a variable-free syntax designed for expressing knowledge about structured concepts and relations between them. They found their

[1] This is typically the case however it is easy to imagine DLs that are undecidable or not fragments of FOL. An example of the latter is languages with transitive closure of roles [64].

primary application in designing *ontologies*, conceptual structures for modeling domain knowledge, and reasoning about them.

1.1.1 Description Logic: From Semantic Nets to OWL 2

Modeling background knowledge has occupied one of the central places in AI since 60s. The approaches can be roughly classified into two major categories: those using FOL as a formal tool enabling automated reasoning, and other systems which often offer intuitively understandable object-oriented representation but lack formal semantics. Prime examples of the latter are *semantic nets* [35] and, later, *frames* [148]. The categories are in some sense complementary: strengths of one approach correspond to weaknesses of the other and vice versa.

DLs emerged in the 80s as a bridge for taking the best from both worlds. It was realized that giving formal semantics to semantic nets and frames was possible without necessity to use complete proof systems for FOL. The first system which proposed a DL-like language with a FOL-like semantics was the famous KL-ONE [23]. It suggested that the language is to the used for controlling domain terminology—which is still one of the central use cases for DL—thus sticking the name "terminological languages" to early DLs.

One especially attractive feature of semantic nets is visualization. The knowledge is represented as a graphical conceptual model with nodes and arcs. For example, one may say that ductal and lobular breast cancers are kinds of cancer by drawing the diagram shown on Figure 1.1. This seems to be more illustrative than a set of formulas in FOL. The syntax of DLs, however, allowed for expressing structured concepts and axioms in a form that was easier to analyze in order to reconstruct the diagram than FOL theories. An example of such DL theory is shown on Figure 1.2. Informally, the axioms state that: breast cancer is cancer that occurs in a part of breast, ductal and lobular cancers are cancers that occur in ducts or lobules respectively, and that ducts and lobules are parts of breast. This knowledge *entails* that ductal and lobular cancers are kinds of breast cancer.

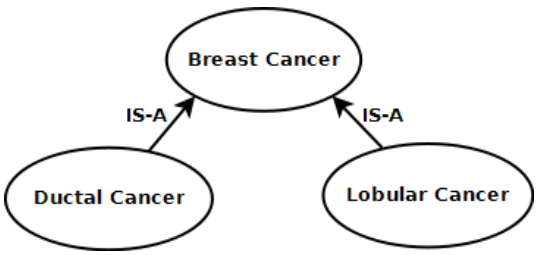

Figure 1.1: Ductal and lobular cancers are kinds of breast cancer.

$$DuctalCancer \equiv Cancer \sqcap \exists occursIn.Duct,$$
$$LobularCancer \equiv Cancer \sqcap \exists occursIn.Lobule,$$
$$Duct \sqsubseteq \exists partOf.Breast,$$
$$Lobule \sqsubseteq \exists partOf.Breast,$$
$$BreastCancer \equiv Cancer \sqcap \exists occursIn.\exists partOf.Breast$$

Figure 1.2: DL axioms which entail that ductal and lobular cancers are kinds of breast cancer.

Early DL systems used so called *structural subsumption* algorithms to derive new knowledge from explicitly represented. Such systems included LOOM [142], KRYPTON [21], CLASSIC [19], and some others. They were computationally tractable but often *incomplete* for many DLs, i.e., not all true statements could be derived. The next generation of systems started to appear in the early 90s with emergence of *tableaux algorithms* [174], which were complete proof systems for DLs. Unfortunately, it turned out that complete reasoning in propositionally closed DLs is PSPACE-complete [152]. However, as is often the case, theoretical complexity was not the last word. The first DL reasoners, which demonstrated acceptable performance on real knowledge bases (KBs) were KRIS [10], FaCT [87], and HAM-ALC [73]. They were followed by modern, highly optimized tableau reasoners, namely, FaCT++ [182], Pellet [177], RACER [72], and HermiT [176].

The success of ontologies in a range of disciplines (see the next section) instigated standardization work on an ontology language. The growing popularity of DL made it the primary candidate for the logical basis of such a language. Two separate developments focused on OIL (Ontology Inference Layer) and DAML (DARPA Agent Markup Language) eventually led to the standardization of OWL (Web Ontology Language) under W3C. OWL originally appeared as three languages (OWL Lite, OWL DL, and OWL Full), where the first two correspond to DLs $\mathcal{SHIF}$(D) and $\mathcal{SHOIN}$(D) respectively. The design and the chosen trade-off between expressivity and reasoning complexity were not flawless and it took several more years to standardize OWL 2. It comes as several profiles, each of which corresponds to a carefully selected DL (see Section 2.1.3).

1.1.2 Ontologies at Work

Ontologies have been remarkably successful in a variety of areas among which the most noticeable are Bioinformatics, Health Care and Life Science (HCLS), and the Semantic Web. We do not intend to give a comprehensive overview of these areas and encourage interested readers to consult the Description Logic Handbook [9].

Ontologies in Bioinformatics, Health Care and Life Science Ontologies are currently most actively used to manage large terminologies in biology, chemistry, and medicine. Bright examples include such large medical ontologies as GALEN, SNOMED CT, and NCI Thesaurus. Their applications range from electronic medical records to data integration and image annotation systems (see, e.g., [43]). The next version of the International Classification of Diseases[2] (ICD-11), the main source of medical codes for classifying symptoms, findings, causes of death, etc., is currently being developed using

[2] http://www.who.int/classifications/icd/en/

OWL.

Biologists and chemists use ontologies to describe their data, in particular, the results of experiments. Notable ontologies include the Gene Ontology (GO), Protein Ontology (PO), Sequence Ontology (SO) and others available through large repositories such as NCBO BioPortal.[3] Several collaborative experiments to create shared science-based ontologies are under way, for example, the Open Biological and Biomedical Ontologies (OBO) Foundry.[4]

Ontologies on the Semantic Web The Semantic Web [17] is a concept of Web in which data is semantically enriched in order to be accessible to machines. OWL ontologies are vital for the success of Semantic Web since they are the principal sources of machine processable semantics. One way of semantic enrichment is through annotations linking content inside HTML documents to relevant terms in ontologies. RDFa is a W3C Recommendation enabling such annotations.

Ontologies are also instrumental for building *semantic Web services* which enable client-server and server-server interactions in an automated fashion. Ontologies, such as OWL-S,[5] have been used to describe semantic services, which allows for their automatic discovery and matchmaking. Another example is the WSMO (Web Service Modeling Ontology),[6] which provides support for deployment and interoperability of services on the Semantic Web.

1.2 Uncertainty

This thesis is especially concerned with one particular limitation of FOL (and, therefore, DLs): its restricted capability of representing various forms of uncertainty. In KR applications, fragments of FOL are typically used to model some abstractions of real world by capturing statements that can be assumed to be crisp in certain. However, as we discuss next, this can be too restrictive.

1.2.1 Uncertainty: Ubiquitous and Versatile

Uncertainty is ubiquitous. It comes in a variety of forms and flavors but before discussing them it is worth noting that FOL does provide some limited support for capturing, but not measuring uncertainty. Examples include disjunction, the Open World Assumption (OWA), and the lack of the Unique Name Assumption (UNA). The OWA allows for dealing with *incompleteness* of knowledge by rejecting *negation as failure*

[3] `http://bioportal.bioontology.org/`
[4] `http://www.obofoundry.org/`
[5] `http://www.w3.org/Submission/OWL-S/`
[6] `http://www.wsmo.org/`

according to each everything which is not provably true is false. The UNA means that every constant in the signature of a FOL knowledge base denotes a distinct domain object. Its rejection help to capture uncertainty about identity of objects, i.e., two constants can denote the same object.

However, there is a whole range of other forms of uncertainty, some of which we briefly present below:

- *Statistics.* A lot of knowledge is obtained through experiments and statistical hypothesis testing. It is often the case that the results come with a mean probability, confidence interval and, perhaps, some information about the distribution. An example could be the statement that "10.4% of cancer incidence among women worldwide are attributed to breast cancer".

- *Beliefs.* While statistical statements usually have frequentist's nature (informally speaking, the results are based on sampling and counting) one may have uncertain judgments about a particular object. An example is the statement "the chances that Mary has breast cancer are 25%". While this statement could have been derived by applying some relevant statistics to Mary, it ultimately reflects one's subjective degree of belief in some propety of Mary's, namely, that she has cancer.

- *Vagueness.* A lot of concepts in human knowledge do not have universally accepted, crisp definitions. Examples are such concepts are "Young" or "Tall". Its characteristic feature is that an object can simultaneously be regarded as a member of the concept and its complement (i.e., young and old, but to a different degree).

- *Subjectivity and Ambiguity.* A related phenomenon is subjectivity, i.e., when the context is determined by some personal perspective. For example, young people tend to have a lower threshold on who is to be considered "young". Ambiguity also refers to the possibility of more than one interpretation of a term, for example, the term "Washington" may refer to a person, a university, the city, or the state.

- *Measurement Errors.* Finally, uncertainty may arise as a result of imperfect measurements. In fact, errors always accrue during measurements or sampling but it is up to the application of whether to account for them.

Statistics, beliefs, and vague knowledge have received most research attention in the context of ontologies. It is important to understand the difference between the nature of the first two forms (which we from now on will call just "uncertainty") versus the third in order to select the right mathematical apparatus.

1.2.2 Uncertainty vs. Vagueness

Uncertainty and vagueness are important in ontologies because they are directly applicable to the terms (concepts and relations) in an ontology. As mentioned above, concepts can be inherently vague and may also be engaged in statistical relationships. Statistics is especially important in areas concerned with drawing conclusions from data (i.e., statistical inference), such as biology, chemistry, and medicine, while vagueness is intrinsic in, for example, natural language processing. Incidentally, ontologies are heavily used in all these areas, so it is no surprise that methods for accommodating these kinds of knowledge have been actively investigated. Two main families of extended DLs have been proposed; one is based on combining DLs with probability theory [92, 82, 119, 52, 38, 139, 70] and the other with the theory of fuzzy sets (see esp. [180, 179] among many others).

The key difference between uncertainty and vagueness is that the former is caused by lack of information while the latter is caused by some inherent imprecision in some term's description. For example, every woman will either develop breast cancer or not and it is only due to lack of information that we cannot determine the outcome for a particular person. On the other hand, we may have arbitrary precise data about one's age and still be unable to say with 100% confidence if the person is young or not. In other words, statements under uncertainty still have Boolean truth values whereas vague statements admit degrees of truth. From a set-theoretic perspective uncertain concepts (resp. uncertain relationsships among concepts) are most reasonably modeled as classical sets (resp. classical relations) but with probability distributions over some outcome space, whereas vague concepts should rather be modeled as fuzzy sets.

This thesis is only concerned with uncertainty and is, therefore, complementary to any work on fuzzy ontologies or fuzzy DLs. The two families of approaches can be partly merged using combinations of probability and fuzzy sets (see [137]) to handle both uncertainty and vagueness.

1.3 P-$\mathcal{SROIQ}$ and Pronto Reasoner

In this thesis we concentrate on a particular formalism, called P-$\mathcal{SROIQ}$ [136], which augments DL $\mathcal{SROIQ}$ (the logical basis of OWL 2) with probabilistic statements. P-$\mathcal{SROIQ}$ has a few features which distinguish it from numerous alternative probabilistic DLs.

- *Support of any OWL ontology.* P-$\mathcal{SROIQ}$ allows for adding probabilistic statements to *any* OWL DL ontology. Most importantly, this means that an already deployed ontology does not have to be translated or modified in any way to serve

as a basis for future probabilistic KBs. In addition, one may represent conditional probabilistic relationships between arbitrarily complex OWL concepts.

- *Purely logical.* P-$\mathcal{SROIQ}$ has model-theoretic semantics in the spirit of FOL and DL. Its axioms work as constraints on models and no graphical inference layer is required.

- *Tight integration.* The classical and the probabilistic parts of a P-$\mathcal{SROIQ}$ KB are inherently semantically connected, which helps to maintain consistency between classical and probabilistic knowledge.

- *Default reasoning.* P-$\mathcal{SROIQ}$ provides a mechanism for non-monotonic (default) inference. It allows for consistent treatment of exceptions as well as applying general probabilistic knowledge to specific objects.

Probabilistic KBs in P-$\mathcal{SROIQ}$ have the structure shown in Figure 1.3. They are composed of a "normal" OWL ontology (classical part of the KB), a set of general probabilistic statements linking concepts from the classical part (probabilistic TBox or PTBox), and a set of probabilistic ABoxes (PABox) which store probabilistic facts about specific objects. In this example the classical part could be a cancer ontology which states, for instance, that breast cancer is a kind of cancer. The PTBox contains two conditional statements informally saying that "given that a random object is a woman, she will develop breast cancer in her lifetime with 0.13 probability" and "given that a random object has the BRCA1 gene mutation she will develop breast cancer with probability between 0.6 and 0.8". Finally, the PABox for Mary contains the unconditional statement that "she has the BRCA1 gene mutation with probability of 0.7–0.8". The logic allows for combining these different pieces of information to infer both general and individual probabilistic knowledge.

P-$\mathcal{SROIQ}$ can be regarded as a representative of the Nilsson-style family of probabilistic logics which date back to Nilsson [156], Hailperin [75], and even George Boole. Their distinctive feature is that probabilistic formulas do *not* determine a unique probability distribution (model), like in Bayesian networks, but rather constrain the set of satisfying distributions. They allow users to specify as much or as little probabilistic knowledge as is available which is again in sharp contrast to Bayesian networks, where conditional probability tables must be complete at every node.

Pronto is the first P-$\mathcal{SROIQ}$ reasoner that scales to hundreds of probabilistic statements [112]. Throughout this thesis it will be our main tool to investigate implementations of various reasoning tasks, including non-monotonic ones. It is currently based on Pellet but, in principle, can work on top of any sound and complete $\mathcal{SROIQ}$ reasoner.

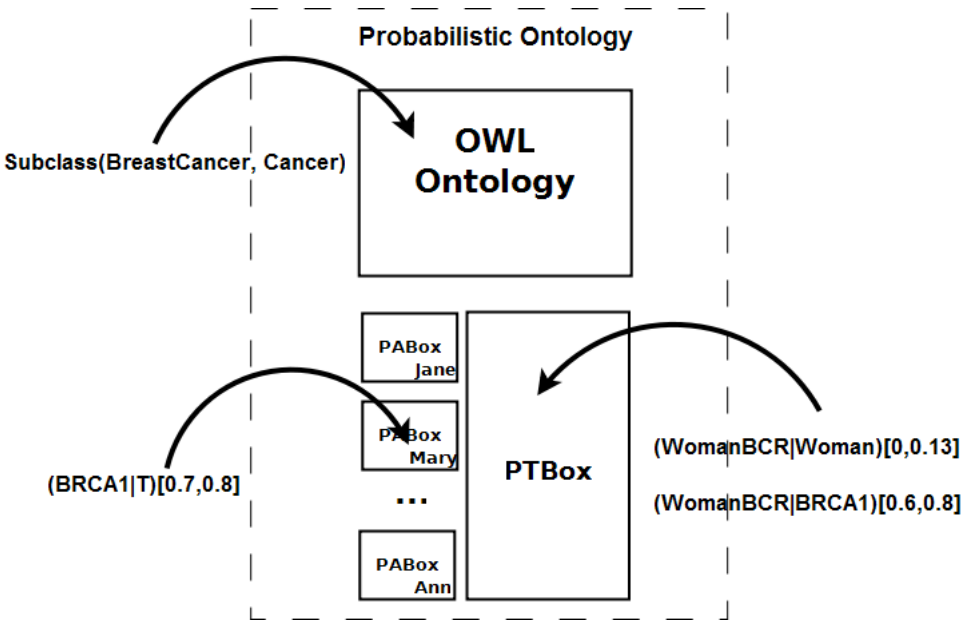

Figure 1.3: The structure of a P-$\mathcal{SROIQ}$ knowledge base (probabilistic ontology).

1.4 Objectives

Nilsson-style probabilistic logics have been criticized on several grounds, most prominently, on the ground of highly impractical reasoning procedures. Due to the lack of independence and other simplifying assumptions, reasoning algorithms, to be complete, have to be global, i.e., process the entire probabilistic knowledge base. Our *first* (and the foremost) goal is to show that reasoning in P-$\mathcal{SROIQ}$ and similar logics, in particular, checking probabilistic satisfiability (PSAT), can still be practical. By "practical" we mean that our implementation can *robustly* solve PSAT for KBs of hundreds of probabilistic statements defined over various OWL ontologies, including large, well-known ontologies which are actively used in real applications.

Our *second* goal is to investigate whether P-$\mathcal{SROIQ}$ could be a suitable language for probabilistic ontologies. It is important to understand whether computational issues are the only obstacles and, if not, how substantial are limitations of P-$\mathcal{SROIQ}$ from a theoretical point of view. Finally, we study several application areas in which P-$\mathcal{SROIQ}$ can be used in its current form and evaluate its utility in those areas.

1.5 Thesis Structure

We now give a brief guide to this thesis. Some of its chapters present already known information while others describe our contributions ranging from application studies and theoretical research to optimization development and evaluation.

Following this introduction, Chapter 2 presents background information covering foundations of DLs, overview of two different families of probabilistic formalisms, namely, Knowledge Base Model Construction (KBMC) and probabilistic logics, preliminaries on P-$\mathcal{SROIQ}$, and related work. Key to understanding the contributions of the thesis are those sections describing DLs (Section 2.1), first-order logics of probability (Section 2.2.2), P-$\mathcal{SROIQ}$ (Section 2.3), and previous work on propositional probabilistic satisfiability (Section 2.4). On the other hand, Section 2.2.1 (and, to some extent, Section 2.2.2) mostly serves the purpose of putting this thesis into a broader context.

The reader interested in practical applications of P-$\mathcal{SROIQ}$ can find examples in Chapter 3. Those include modeling uncertain knowledge about breast cancer, finding inconsistencies in a medical expert system, and probabilistic validation of ontology alignments. Note that not all our conclusions are positive, for example, a number of challenges are described in Section 3.1.

Chapters 4 and 5 present the core contributions of this thesis: a theoretical analysis of P-$\mathcal{SROIQ}$ and optimized reasoning algorithms respectively. The analysis is based on a translation of P-$\mathcal{SROIQ}$ into a first-order logic of probability similarly to how standard DLs are understood as fragments of FOL. The translation helps to explain certain limitations of P-$\mathcal{SROIQ}$ and motivates the effort to address them. In the algorithms chapter, the central role is played by the PSAT procedure (Section 5.1) which is the first scalable algorithm for deciding non-propositional PSAT. Novel algorithms for other reasoning problems are also presented.

The next two chapters are dedicated to an extensive evaluation of the algorithms. Unfortunately, almost no naturally occurring probabilistic ontologies existed prior to our work so their generation for evaluation purposes was a substantial challenge by itself. The generation methodology and the results of the synthetic evaluation of the PSAT algorithm are presented in Chapter 6 while Chapter 7 describes the evaluation of the algorithms for Diagnosis, consistency, and non-monotonic entailment problems.

Chapter 8 is a brief system description of Pronto while Chapter 9 concludes the thesis with a review of contributions, summary of outstanding issues, and suggestions for future research in the area.

Chapter 2

Background and Related Work

This chapter presents background information most of which is necessary for understanding later chapters. Preliminaries are presented in the first three sections which describe classical Description Logics, a set of formalisms that have been designed to represent and reason about uncertainty, and, eventually, P-$\mathcal{SROIQ}$, which is of the main interest in this thesis. Finally, the chapter also includes information on related work focused on implementation of reasoning algorithms for probabilistic logics, in particular, propositional PSAT algorithms. Since the body of related work, especially on practical and implementable reasoning procedures, is not too extensive, we decided not to take it into a dedicated chapter.

2.1 Description Logic

Description Logics (DLs) form a family of logics which are typically decidable fragments of first-order logic developed specifically for representing structural background knowledge [9]. $\mathcal{SROIQ}$ is one of the most expressive representatives of that family. It is the formal basis of the Web Ontology Language (OWL 2) which is a W3C standard for representing ontologies.

2.1.1 Syntax and Semantics

This section briefly presents $\mathcal{SROIQ}$. We omit some technical details which are not essential for later chapters. The full presentation of $\mathcal{SROIQ}$ can be found in [88].

Syntax of $\mathcal{SROIQ}$

We assume fixed finite sets N_C, N_R, and N_I of concept names (atomic concepts), role names, and individuals, respectively. N_C is assumed to contain the special names $\top$ and $\bot$ while N_R contains the universal role name U. The set of *roles* is defined as

$N_R \cup \{R^- \mid R \in N_R\}$. Additionally, we define the function Inv such that $Inv(R) = R^-$ and $Inv(R^-) = R$. We start with defining the syntax of role and concept expressions:

Definition 2.1 (Role Chains and Concept Expressions)**.** *A **role chain** is a finite sequence of names $R_1 \dots R_k$ where $R_i \in N_R$. **Complex concepts** (concept expressions or just concepts) are expressions generated by the following grammar:*

$$C ::= A \mid \{o\} \mid \neg C \mid C \sqcap D \mid \exists R.C \mid \exists S.\text{Self} \mid \geq nS.C \mid \leq nS.C$$

where $A \in N_C, S \in N_R$, R is a role, C and D are concepts, n is a natural number, $o \in N_I$. Expressions of the form $C \sqcup D$, $\forall R.C$, $\{o_1, \dots, o_k\}$, and $= nR.C$ abbreviate $\neg(\neg C \sqcap \neg D)$, $\neg\exists R.\neg C$, $\{o_1\} \sqcup \dots \sqcup \{o_k\}$, and $\geq nR.C \sqcap \leq nR.C$, respectively. Abbreviations of the form $\{o_1, \dots, o_k\}$ are called nominal expressions.

A knowledge base (or *ontology*) in $\mathcal{SROIQ}$ is a tuple $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{R})$ where $\mathcal{T}$ is a terminological box (TBox), $\mathcal{A}$ is an assertional box (ABox[1]) and $\mathcal{R}$ is a role box (RBox). A TBox $\mathcal{T}$ is a finite set of *general concept inclusion* axioms (GCIs), an ABox $\mathcal{A}$ is a finite set of *individual assertions*, and an RBox $\mathcal{R}$ is the union of a *role hierarchy* and a finite set of *role assertion* axioms.[2] GCIs are also commonly referred to as concept subsumptions or simply *subsumptions*.

Definition 2.2 (GCI, Individual Assertions, Role Hierarchy, and Role Assertions)**.** *A **general concept inclusion** axiom is an expression of the form $C \sqsubseteq D$ where C and D are (possibly complex) concepts. $C \equiv D$ abbreviates $\{C \sqsubseteq D, D \sqsubseteq C\}$. An **individual assertion** is an expression of one of the following forms: $a : C$, $(a, b) : R$, $(a, b) : \neg R$, or $a \neq b$, where $a, b \in N_I$, C is a (possibly complex) concept, and R is a (possibly inverse) role. A **role hierarchy** $\mathcal{R}_h$ is a finite set of role inclusion axioms which are expressions of the form $\omega \sqsubseteq S$ where $S \in N_R$ and ω is a role chain. The syntax of **role assertions** has the following form [88]:*

$$RA ::= \text{Sym}(R) \mid \text{Tra}(R) \mid \text{Ref}(R) \mid \text{Irr}(R) \mid \text{Dis}(R, S)$$

[1] In what follows we will generally neglect ABoxes when dealing with $\mathcal{SROIQ}$ ontologies because nominals permit us to reduce $\mathcal{SROIQ}$ ABoxes to $\mathcal{SROIQ}$ TBoxes [183]. For example, $a : C$ can be expressed as $\{a\} \sqsubseteq C$ and $(a, b) : R$ as $\{a\} \sqsubseteq \exists R.\{b\}$.

[2] Note that, in general, arbitrary RBoxes are not properly part of $\mathcal{SROIQ}$ since they lead to undecidability rather easily. Typically, a rather complex set of syntactic conditions (over both the TBox and RBox) are imposed to ensure decidability (and the standard complexity bounds). For example, transitive roles are not allowed to occur in number restrictions. See [88, 106] as well as `http://www.w3.org/TR/2009/REC-owl2-syntax-20091027/#Global_Restrictions_on_Axioms_in_OWL_2_DL`. Note that for our purposes, the exact restrictions are immaterial as long as they ensure decidability and a given complexity.

where R and S are roles not equal to U.

Informally, concept and role inclusion axioms specify that one, possibly complex, concept (or a role) is a sub-concept (or a sub-role) of another. The expressions on the right hand-side of role assertion axioms stand for symmetric, transitive, reflexive, irreflexive, and disjoint roles respectively. We now turn to a formal presentation of the semantics.

Semantics of $\mathcal{SROIQ}$

Semantics of DLs is standardly based on interpretations $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is a non-empty set (the domain) and $\cdot^{\mathcal{I}}$ is an interpretation function that maps each $A \in N_C$ to a subset $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, each $R \in N_R$ to a relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ and each $o \in N_I$ to an element $o^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. Inverse roles are interpreted as inverse relations, i.e., for each $R \in N_R$ we have

$$(R^-)^{\mathcal{I}} = \{(y, x) \mid (x, y) \in R^{\mathcal{I}}\}.$$

The interpretation of U is fixed to $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. Clearly $(U^-)^{\mathcal{I}} = U^{\mathcal{I}}$.

Next we extend the interpretation function to concept expressions in the following way (in the next two definitions S is a role name, R is a role, C and D are concepts, n is a natural number, o is an individual name, and $\sharp M$ stands for the cardinality of M):

Definition 2.3 (Concept Semantics)**.**

$$\begin{aligned}
&\top^{\mathcal{I}} = \Delta^{\mathcal{I}} \\
&\bot^{\mathcal{I}} = \emptyset \\
&(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
&(\{o\})^{\mathcal{I}} = o^{\mathcal{I}} \\
&(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
&(\exists R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \exists y \in \mathcal{C}^{\mathcal{I}} \ s.t. \ (x, y) \in R^{\mathcal{I}}\} \\
&(\exists S.\mathrm{Self})^{\mathcal{I}} = \{x \mid (x, x) \in S^{\mathcal{I}}\} \\
&(\geq nS.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \sharp\{y \in \mathcal{C}^{\mathcal{I}} \ s.t. \ (x, y) \in R^{\mathcal{I}}\} \geq n\} \\
&(\leq nS.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \sharp\{y \in \mathcal{C}^{\mathcal{I}} \ s.t. \ (x, y) \in R^{\mathcal{I}}\} \leq n\}
\end{aligned}$$

Next we define the "satisfies" relation, denoted as $\models$, between interpretations and different kinds of axioms in $\mathcal{SROIQ}$:

Definition 2.4 (Axiom Semantics)**.**

Concept inclusion:	$\mathcal{I} \models C \sqsubseteq D$ *if* $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
Role inclusion:	$\mathcal{I} \models R_1 \dots R_k \sqsubseteq S$ *if* $R_1^{\mathcal{I}} \circ \dots \circ R_k^{\mathcal{I}} \subseteq S^{\mathcal{I}}$
Role assertions:	$\mathcal{I} \models \mathrm{Sym}(R)$ *if* $(x,y) \in R^{\mathcal{I}} \Rightarrow (y,x) \in R^{\mathcal{I}}$
	$\mathcal{I} \models \mathrm{Tra}(R)$ *if* $(x,y) \in R^{\mathcal{I}}$ *and* $(y,z) \in R^{\mathcal{I}} \Rightarrow (x,z) \in R^{\mathcal{I}}$
	$\mathcal{I} \models \mathrm{Ref}(R)$ *if* $\{(x,x) \mid x \in \Delta^{\mathcal{I}}\} \subseteq R^{\mathcal{I}}$
	$\mathcal{I} \models \mathrm{Irr}(R)$ *if* $\{(x,x) \mid x \in \Delta^{\mathcal{I}}\} \cap R^{\mathcal{I}} = \emptyset$
	$\mathcal{I} \models \mathrm{Dis}(R_1, R_2)$ *if* $R_1^{\mathcal{I}} \cap R_2^{\mathcal{I}} = \emptyset$

An interpretation satisfies (or is a model of) a TBox $\mathcal{T}$ (resp. an RBox $\mathcal{R}$), denoted as $\mathcal{I} \models \mathcal{T}$ (resp. $\mathcal{I} \models \mathcal{R}$), if $\mathcal{I}$ satisfies all axioms in $\mathcal{T}$ (resp. $\mathcal{R}$). It satisfies (or is a model of) a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{R})$, denoted as $\mathcal{I} \models \mathcal{K}$, if it is a model of both $\mathcal{T}$ and $\mathcal{R}$.

2.1.2 Reasoning Problems and Complexity

There are several standard reasoning problems for DLs, including $\mathcal{SROIQ}$ [9]:

Knowledge Base Satisfiability (SAT) Given a knowledge base $\mathcal{K}$ determine whether there exists an interpretation $\mathcal{I}$ such that $\mathcal{I} \models \mathcal{K}$.

Concept Satisfiability (CSAT) Given a knowledge base $\mathcal{K}$ and a (possibly complex) concept C determine whether there exists an interpretation $\mathcal{I}$ such that $\mathcal{I} \models \mathcal{K}$ and $C^{\mathcal{I}} \neq \emptyset$.

Entailment (ENT) Given a knowledge base $\mathcal{K}$ and an axiom η determine whether $\mathcal{I} \models \eta$ for all $\mathcal{I} \models \mathcal{K}$.

These problems are pairwise reducible. In particular, an instance of CSAT can be solved by adding the new axiom $\{a\} \sqsubseteq C$ in $\mathcal{T}$, where a is an individual name not occurring in $\mathcal{K}$, and then solving SAT. An instance of ENT where η is a TBox axiom, i.e., $\eta = C \sqsubseteq D$, can be reduced to SAT by adding the axiom $\{a\} \sqsubseteq C \sqcap \neg D$, where a is also a fresh name. Since the reductions are polynomial the CSAT and ENT problems belong to the same complexity class, which has been determined to be `co-N2ExpTime`-complete (since SAT is `N2ExpTime`-complete) [106].

2.1.3 The Web Ontology Language

The Web Ontology Language (OWL) is a standardized language for defining ontologies on the Semantic Web.[3] It has direct connections to DLs: in essence, OWL can be thought of as a combination of a DL, datatypes for managing data assertions (such as measurements), a set of syntactic forms most of which are based on XML, and several non-logical features, e.g., annotations.[4] The current version of OWL is OWL 2.

Syntactically, an OWL ontology is a document conforming to one of the specified syntaxes, i.e., RDF/XML, OWL/XML, Manchester syntax, etc. Semantically, it is a DL knowledge base, so reasoning tasks in OWL can be reduced to SAT in DL [89]. OWL 2 comes as a set of so called *profiles* each of which embodies a trade-off between expressivity and the computational complexity of reasoning. OWL EL corresponds to the logic $\mathcal{EL}^{++}$ [8]. OWL QL is based on the logic DL-Lite [26]. The standard reasoning tasks for OWL EL and QL are polynomial. OWL RL is inspired by "rule based" DLs such as Description Logic Programs (DLP) [71]. OWL (2) DL corresponds to $\mathcal{SROIQ}$ and is the most expressive decidable language in the OWL family. OWL 2 Full, in turn, is an undecidable language which allows interpretation of any well-formed RDF document as an OWL ontology.

Due to the direct connection between OWL 2 DL and $\mathcal{SROIQ}$ our work can be regarded as an attempt to develop practical algorithms for reasoning with *probabilistic ontologies* (i.e., OWL ontologies with probabilities). Indeed, as will be shown in Chapter 8, we use meta-logical features of OWL 2, namely annotations, to attach probabilities to OWL axioms. Such axioms are then mapped to probabilistic statements in P-$\mathcal{SROIQ}$ to form knowledge bases in probabilistic DL. We generally only discuss $\mathcal{SROIQ}$ and its close relatives, but our results also apply to its sub languages, e.g., $\mathcal{EL}^{++}$ or DL-Lite. However, we do not attempt to take advantage of any of the special properties of various fragments, such as completeness of polynomial-time reasoning in $\mathcal{EL}^{++}$, which might provide for distinct sorts of optimization.

2.2 Representation and Reasoning About Uncertainty

The challenge of developing expressive knowledge representation and reasoning formalisms that are capable of dealing with various sorts of uncertainty is long standing. This issue is one of the central in AI and is also of considerable interest in health care and life science community. It has been recognized that classical graphical models, such

[3] It is widely claimed to be a Semantic Web language, especially in W3C normative documents, although use cases for OWL go far beyond what was traditionally believed to be the Semantic Web, i.e., the Web of semantically rich data.

[4] A complete OWL reference can be found in normative documents listed at `http://www.w3.org/TR/owl2-overview/`.

as Bayesian or Markov networks, despite offering a clear semantics and reasonably tractable inference algorithms, are often inadequate for modeling in complex domains because of being essentially propositional. Consequently, a lot of research went into developing formalisms combining the expressive power of first-order logic with the ability to represent and reason about probabilistic statements.[5]

We do not aim to present an extensive survey of first-order probabilistic languages that have emerged over the last two decades (the interested reader is encouraged to see [147, 44]). Instead, we describe two substantially different approaches which capture two fundamental approaches to combining probabilities with logic, namely, Knowledge Based Model Construction (KBMC) formalisms and first-order logics of probability (FOPLs). They differ in several key aspects, in particular, the former tend to provide users with tools to specify the unique probability distribution over relational structures while the latter only constrain the set of probability distribution in the same spirit as formulas in first-order logic place restriction on the set of models [147]. Furthermore, KBMC formalisms typically result from attempts to increase expressive power of graphical models while FOPL are extensions of first-order logic to accommodate probabilistic knowledge. This distinction is important because both kinds of formalisms tend to inherit advantages and shortcomings of their roots. KBMC formalisms often provide reasonably efficient inference algorithms, modular knowledge representation, and the ability to draw entailments from the unique distribution, but on the other hand they require users to *fully* describe the model. FOPL, in turn, enable users to specify no more information than they have, are more understandable to people having worked with standard FOL or its fragments (e.g., DLs), but suffer from two major drawbacks: inferential weakness (due to the lack of a unique distribution) and very high computational complexity or even undecidability.

FOPL are closer related to the work presented in this thesis so they will be covered in more detail in Section 2.2.2. We will show in Chapter 4 that P-$\mathcal{SROIQ}$ can be thought of as a particular instance of FOPL and can further evolve by building on work that has been done in the context of FOPL. However, before we move to FOPL we briefly present the KBMC branch of probabilistic languages which have been influential in many fields and recently started to expand in the area of the Semantic Web [39, 49].

2.2.1 Knowledge Based Model Construction

The KBMC family of languages provide rich syntaxes to describe complex conceptual models with uncertainty. They provide a first-order declarative language thus allowing users to compactly specify a template for probability distributions over complex data

[5] We largely ignore other theories of uncertainty besides probability—Dempster-Shafer theory of evidence, fuzzy logic, and the possibility theory among others—in this thesis.

spaces, e.g., a relational or object database, or possible Herbrand interpretations. Most KBMC formalisms define distributions over relational structures, or *possible worlds*, which are constrained by available background knowledge. Inference, such as query answering, is typically done by generating a propositional graphical model (Bayesian or Markov network) from a first-order specification. This idea dates back to early works of Horsch and Poole [90] and Breese [24, 188] and eventually led to a wealth of languages some of which are briefly characterized below.

Probabilistic Relational Models

We start with Probabilistic Relational Models (PRM)—a classical KBMC formalism which extends traditional Bayesian networks by providing means to describe objects, their attributes, and links between them [57, 121, 63]. It is "to Bayesian networks what relational logic is to propositional logic" [63]. A PRM model is a template for specifying a unique probability distribution over the set of possible instantiations of a relational schema (possible worlds). It can handle different kinds of uncertainty: *attribute uncertainty* (uncertainty about values of object's attributes) and *structural uncertainty* (uncertainty about existence of relations between objects).

PRM is composed of two sub-languages: the first is used to specify logical schemas (relational models) while the second defines a graphical template for various probabilistic links between domain objects (probabilistic relational models). The schema language is less expressive than first-order logic and corresponds to the language of relational databases. Schema description consists of a set of *classes* (or types of objects) and a set of *relations*. Each class is characterized by a set of attributes whose values are drawn from a fixed domain. Objects can refer other objects via typed reference slots which correspond to foreign keys in databases. An *instance* of a schema is a relational logic interpretation of the schema's entities (classes and relations). It contains objects for each class and a boolean function which specifies which relations hold. Importantly, an instance can be specified partly via a *relational skeleton* which describes all objects and relations but leaves out some attributes. Intuitively, missing attributes are those whose values are uncertain.

The probabilistic model language provides a syntax for describing qualitative dependency structures and their numerical parameters. The structure specifies how values of uncertain attributes in a relational skeleton depend on other attributes in the same skeleton. Each attribute has a set of parents, attributes of the same object or attributes of objects referenced via slots, that collectively determine its value. Such structure is naturally modeled using Bayesian networks where each attribute corresponds to a node. Note, however, that the networks define the structure on the class level thus avoiding the need to duplicate the dependencies for each object of the same class. Structure

parameters are conditional probability distributions (CPD) that induce a global probability distribution over possible completions of the skeleton (i.e., over schema instances consistent with the skeleton). Coherence of global distributions has been proved subject to acyclicity conditions on dependency structures. RPM supports the notion of *aggregation functions* which enable modeling of complex dependencies, for example, the dependency of a student's average grade on the average number of course (s)he enrolls in.

Exact and approximate inference algorithms have been developed for PRM. All of them presume that the ground Bayesian network has been generated from the relational skeleton and the probabilistic model.[6] Exact algorithms are used for skeletons which lead to either small or well-structured Bayesian networks. Approximate algorithms, e.g. approximate belief propagation [163], are used for large networks. Empirical evidence suggests that such algorithms often converge to correct marginal probabilities for the value of each attribute even in general networks, i.e., those with more than one path between nodes [144].

Relational Bayesian Networks

Relational Bayesian Networks (RBN) [93] are similar to PRM since they are also allow for Bayesian modeling on the predicate level. The essential difference is that the formalism does not consist of separate components for specifying schemas and probabilistic dependencies. Every RBN model is a Bayesian network where each node stands for a predicate name in the vocabulary. Given the predicate r, its node N_r acts as a random variable whose set of values is the set of all groundings of r with respect to a finite domain. Extensions of RBN for reasoning about infinite domains were developed later but at the cost of restricting recursion [94].

Such a representation may appear very inefficient since the extensions of predicates (i.e., the domains of nodes) are not restricted by a relational skeleton as in PRM. Therefore, defining conditional probability distributions is not feasible due to prohibitively large sets of values. However, RBNs are centered around so called *combination functions*, such as the well-known noisy-or [163], which act as first-order representations of CPDs. For given n-ary predicate r a combination function maps instantiations of predicates associated with parent nodes of r to probabilities of instantiations of r. Every node in an RBN is labeled with a combination function along with its predicate name.[7]

[6] Note, that such a network does not always need to be complete. Clearly the size of the complete ground network is exponential in the domain size of missing attributes in the skeleton so full construction is intractable. However, it is often possible to prune segments of the network which are provably irrelevant to the query thus reducing its size.

[7] Technically, nodes are labeled with probability formulas of which combination functions are a special case but such details are unimportant.

Jaeger developed a flexible framework for defining combination functions which allows for their nesting. Furthermore, he allows RBNs to be *recursive* which means that some instantiations of a predicate can depend probabilistically on other instantiations of the same predicate. This is essential for modeling temporal dependencies (where $r(t, x)$ depends on $r(t-1, x)$), symmetric relations ($r(x, y)$ and $r(y, x)$), and stochastic functions which map each combination of arguments to a probability distribution over the set of output values. This feature is very powerful, for example, it makes RBNs more expressive than PRMs, Markov Logic Networks (MLNs) [95, 97] and, due to the ability to model random functions, Object Oriented Bayesian Networks [120]. On the other hand, it complicates the otherwise very transparent semantics of RBNs because recursive dependencies can easily introduce cycles. As a solution Jaeger proposes to impose well-founded orderings on tuples of domain elements. These orderings then act as extra nodes and help to ensure that the network remains acyclic. Unfortunately, this solution works only for finite domains whereas in infinite domains checking well-foundness is undecidable [94]. Similarly to PRM inference in RBNs is done via reasoning on grounded Bayesian networks which are constructed on the fly for each specific query.

Multi-Entity Bayesian Networks

Multi-Entity Bayesian Networks (MEBNs) [126, 125] further increase expressivity of graphical models comparing to RPMs, OOBNs, and RBNs. MEBN is a full first-order generalization of Bayesian networks which is capable of representing a broad class of probability distributions over interpretations of *any* finitely axiomatizable first-order theory. In contrast to many other KBMC languages MEBN is, first, an inherently open world formalism and, second, does not require finiteness of the domain. MEBN has been designed with the goal of achieving maximal expressivity so it can be used as a common formalism for exchanging with probabilistic knowledge [125].[8]

A probabilistic model in MEBN (or a MEBN theory, MTheory) is a collection of so called MEBN fragments (MFrags) each of which is a parametrized Bayesian network involving a small number of variables. MFrags are basic building blocks for creating *modular* probabilistic models in MEBN. An MFrag is similar to a relational Bayesian network but, in addition, can contain *context terms* which place logical restrictions on instantiations of internal random variables. MFrags may share random variables which serve as catenation points for joining a collection of fragments satisfying global consistency constraints into a single coherent theory. Similarly to RBNs, MFrags can express recursive relationships to allow particular instantiations of random variable to depend on other instantiations of itself. MEBN provides standard MFrags with

[8] Similar goals have been pursued by designers of some well known non-probabilistic first-order KR formalisms, for example, Common Logic (ISO/IEC 2007) or Knowledge Interchange Format (KIF).

prescribed semantics for representing logical connectives including quantification which are essential for achieving the expressive power of FOL.

MEBN uses the notion of an *influencing configuration* to specify local probability distributions for random variables in MFrags. An influencing configuration is an instantiation of parent variables which is relevant to the child's distribution. As typical for first-order graphical languages a variable can have different number of parents depending on a particular instantiations. RPMs deal with this issue using aggregation functions while RBNs employ combination functions. MEBN, in turn, allows for complex specifications of local distributions which consist of rules that i) determine influencing configurations and ii) assign probabilities to possible values for each configuration. MEBN's mechanism can express both aggregation and combination functions [125].

Another interesting feature of MEBN not present in RPM or RBN is the separation between generic, or terminological knowledge (TBox) and factual, or assertional knowledge (ABox). The former is represented in *generative* MFrags while the latter is represented in *finding* MFrags. Generative fragments capture knowledge about generic statistical relationships while finding fragment specify information about a concrete situation. Such separation is not new: it dates back to classical KR&R systems for conceptual modeling, such as KRYPTON or KL-ONE [22, 23], and is now a basic feature of DLs and OWL (as described in Section 2.1).[9]

Conceptually, inference in MEBN is done in the spirit of other KBMC formalisms: by reasoning over grounded Bayesian networks. However, it is more involved than inference in PRM or RBN. Since MEBN allows for arbitrary first-order theories the entailment problem is undecidable, so one cannot hope for instantiating a *finite*, even if prohibitively large, Bayesian network. Therefore, the authors use an iterative, anytime algorithm for constructing situation specific Bayesian networks (SSBN [143]) which generates a series of approximate networks whose size is increased at every step. Eventually, either the algorithm terminates or a stopping condition, e.g., time-out, is reached. In the first case the algorithm has either computed exact probabilities for target random variables or detected an inconsistency between findings and the generative part of the theory. In the second case the algorithm has computed an approximate answer. Importantly, the algorithm always terminates on inconsistent theories [125] so it is a refutation-complete procedure, similarly to the classical resolution for FOL.

Markov Logic Networks

Finally, we describe Markov Logic Networks (MLNs)—is a recently developed KBMC formalism which extends a distinct kind of graphical model, Markov networks [109],

[9] As described in the same section, the distinction is blurred in expressive DLs with nominals.

with first-order representational power [170]. Markov networks are undirected models of uncertainty which, similarly to Bayesian networks, represent a unique joint distribution for a set of random variables but, in contrast to them, use weights instead of conditional probabilities. They have pros and cons when compared to Bayesian networks, in particular, they are more generic by allowing cycles but the parameters (weights) are not simply conditional probabilities as in Bayesian case. However, the weights are still uniquely determined by probabilities and can be computed via an optimization algorithm (see, e.g., [170]). This and other differences are present in first-order generalizations of both models.

MLN are similar to MEBN in their aim to generalize first-order logic. Syntactically an MLN is a set of pairs (F_i, w_i), where F_i is a (possibly quantified) formula in FOL and w_i is its weight. The weight is not a probability but can be understood as a *strength* of the formula: the higher the weight, the lower the probability of interpretations which do not satisfy the formula.[10] Analogously to other KBMC formalisms MLNs are templates for ground Markov networks. An instantiation of an MLN contains a clique for every formula wherein the nodes of the clique are ground atoms which appear together in at least one grounding of the formula. Every clique inherits the weight of the original formula.

MLN generalizes deductive inference in first-order logic but, differently from MEBN, under three assumptions: i) the unique name assumption (UNA), ii) the domain closure assumption (the domain is the union of constants' interpretations closed under function applications), and iii) the known functions assumption (KFA) according to which any application of any function maps the arguments to one of the constants. The first two assumptions can be lifted by introducing equality (UNA) and a finite number of unknown domain objects (domain closure). KFA is more restrictive as it precludes infinite Herbrand universes. It can be slightly relaxed by restricting the level of nesting for function applications, otherwise the formalism needs to be extended to deal with infinite models (see [50] on modeling infinite domains using Markov networks).

MLNs face the typical problem of KBMC formalisms: the size of ground networks is exponential in the size of the domain. Furthermore, exact inference in Markov networks is #P-complete, so approximate methods are needed. Differently from MEBN, which compute a series of approximate networks, MLN uses sampling techniques, in particular, Markov Chain Monte-Carlo method. The basic idea is to construct a Markov chain whose equilibrium distribution is the desired joint distribution of the complete ground network. The construction yields an anytime inference algorithm as the quality of the approximation improves at every step, when a new ground atom is sampled from the neighborhood (the Markov blanket) of the formula which probability is being computed.

[10] Technically it is slightly more complicated. The weight corresponds the difference in log probability between satisfying and non-satisfying interpretations (worlds).

Summary

The above list of KBMC formalisms is by no means exhaustive but is sufficient to draw several general conclusions. Languages of this family share the following properties: they are extensions of their corresponding graphical models, they provide means for describing first-order templates for instantiating ground probabilistic models, they describe a unique probability distribution over relational structures, and their principal reasoning mechanism is inference on instantiated ground network. As usual, these properties have advantages and shortcomings.

The biggest advantages are modular knowledge representation, inferential power, and learning support. Underlying graphical models are inherently modular due to the causal Markov property which allows for separating independent fragments of probabilistic theory. This is best seen in MEBN which theories are loosely coupled collections of MFrags, each of which is a first-order theory. Inferential power reflects the ability to make inference from the *unique* probability distribution over possible worlds. Such a distribution assigns a probability to *every* first-order formula thus giving meaningful answers to each query. Finally, first-order models in these formalisms can be learned by using adapted learning algorithms for learning and generalizing Bayesian or Markov networks from data (see esp. [57, 96, 170] on learning RPM, RBN, and MLN models).

However, there are also difficulties with using KBMC formalisms to manage uncertainty in logic-based ontologies, e.g., in the Semantic Web. They are succinctly summarized in a recent survey of probabilistic formalisms for the Semantic Web [166]:

> [KBMC] approaches are rather unsatisfying because they do not consider the semantics of Semantic Web languages but rather focus at a special kind of probabilistic model, i.e. [BNs] or [MEBNs], and provide a Semantic Web based syntactical interchange format for these probabilistic models and their semantics.

The key point is KBMC-based extensions to logical languages are strongly decoupled from the underlying classical logic. For example, a major issue in PR-OWL (a Semantic Web language based on MEBN [39]) is the weak connection between the probabilistic theory (MTheory) and the underlying OWL ontology. Specifically, it is not the terms from the ontology that engage in probabilistic relations but rather random variables which do not (automatically) maintain the terms' semantics. This problem can be rectified by formalizing a mapping between OWL and PR-OWL (see [28]) but it still requires translation of the ontology into MEBN.

Also, these formalisms do not properly support the scenario when probabilistic, e.g., statistical statements are to be added to a large existing OWL ontology such as the NCI Thesaurus. In this case it is highly undesirable to translate the ontology into a new

formalism because it may evolve separately from the probabilistic model (it can be used by applications which do not require uncertainty). Also, the commitment to a unique distribution, while increasing inferential power, often requires unknown probabilistic knowledge to be specified or assume independence where it is not fully appropriate. This goes out of accord with basic principles of designing ontologies using DLs and OWL which never force users to specify more information than is actually known.

2.2.2 Probabilistic Logics

We next proceed to purely logical approaches to probabilistic languages. By "purely logical" we mean that probabilistic statements (formulas or axioms) simply place constraints on probabilistic models without (necessarily) enforcing a single, unique probability distribution, as KBMC formalisms do. In other words, the statements act just as classical logical axioms that constrain the set of satisfying interpretations (truth assignments or first-order structures). We begin with the propositional case and then move on to highly powerful first-order logics of probability.

Probabilistic Propositional Logic

Propositional probabilistic logic (PPL) dates back to the ideas of George Boole and was re-invented and formalized by Hailperin [75] and, famously, Nilsson [156, 157]. It was the first probabilistic logic to use probability distributions over possible worlds (i.e. model structures) rather than over sentences of the formal language (as, e.g., in [59]). The basic Nilsson model presented below is a foundation of many subsequently developed formalisms including probabilistic logic programming [153, 132] and P-$\mathcal{SROIQ}$ [136].

The *syntax* of Nilsson's basic model is quite simple: we assume a finite set of n Boolean variables $\mathcal{X} = \{x_1, \ldots, x_n\}$ and a set $\mathcal{S} = \{s_1, \ldots, s_m\}$ of m propositional formulas constructed from $\mathcal{X}$ by means of standard logical connectives $\wedge, \vee, \neg$. A knowledge base $\mathcal{K}$ in PPL is a set of pairs (s_i, p_i), where $s_i \in \mathcal{S}$ and $p_i \in [0, 1]$ is a real number. Each pair is a *probabilistic formula*, where p_i specifies the probability of s_i.

The semantics of PPL is based on possible worlds, each of which is a truth assignment to all variables in $\mathcal{X}$. Clearly there are 2^n possible worlds for n variables. A probabilistic interpretation Pr is a probability distribution over W, the set of all possible worlds over $\mathcal{X}$. An interpretation Pr assigns each $s_i \in \mathcal{S}$ a real number, $Pr(s_i)$, which is equal to the total probability of all truth assignments $w \in W$ which satisfy s_i (written as $w \models s_i$). Pr satisfies (or is a model of) a probabilistic formula (s_i, p_i) if $Pr(s_i) = p_i$.

The basic reasoning task in PPL is the problem of determining consistency of sets of probabilistic formulas. A set $\mathcal{K}$, a *probabilistic knowledge base*, is called consistent

(or satisfiable) if there exists a probabilistic interpretation that satisfies all probabilistic formulas in $\mathcal{K}$. The problem can be reduced to the problem of solubility of the following instance of linear inequalities over real-valued variables $z_j, j \in \{1, \ldots, 2^n\}$ [156]:

$$\begin{aligned} &\sum_{w \models s_i} z_w = p_i, \text{ where } (s_i, p_i) \in \mathcal{K}, \\ &\sum_{w \in W} z_w = 1, \text{ and } z_w \geq 0 \text{ for every } w \in W \end{aligned} \tag{2.1}$$

Each variable z_w corresponds to probability of the world $w \in W$ in some probabilistic interpretation Pr. The inequalities of (2.1) specify that i) the probability of each sentence s_i is p_i, ii) the total probability of all worlds in W is 1, and iii) the probability of any world must be non-negative. A solution to this system *is* a satisfying probabilistic interpretation of $\mathcal{K}$.

The problem of *entailing* the tight probability bounds for a formula s given $\mathcal{K}$ is defined as the problem of computing a pair of numbers l, u such that l is the minimum (resp. maximum) of $Pr(s)$ over all probabilistic models of $\mathcal{K}$. The numbers can be computed by minimizing (resp. maximizing) the linear expression $\sum_{w \models s} z_w$ subject to the (soluble) system (2.1).

Observe that the number of variables in (2.1) is exponential in the number of variables in $\mathcal{X}$. This is the major computational obstacle to using PPL. Many authors have proposed different approaches to tackle it, which range from using approximate heuristic methods [156] to advanced Linear Programming (LP) techniques [105, 98]. The latter have been particularly successful for PPL. The principal aim of this thesis is to show that the same idea, although requiring a major re-thinking, can lead to practical reasoning in non-propositional cases.

First Order Logics of Probability

Next we describe a family of first-order logics of probability (FOPL) as defined and analyzed by Bacchus, Halpern, and Abadi [12, 76, 1]. To our knowledge, these logics represent the most general formalisms allowing fusion of classical first-order and probabilistic knowledge. They treat probabilities in a natural way on both syntactic and semantic levels, do not make commitments to point-valued, or even quantitative probabilities, and are well suited for representing probabilistic statements of different natures. Finally, the logics can serve as bases for designing systems of default reasoning. These features make them a perfect framework for studying probabilistic extensions of Description Logics, as will be demonstrated in Chapter 4.

Before proceeding to syntax and semantics of the logics in this family we must comment that the logics defined by Bacchus [12] and Halpern [76, 1] differ in their notions of probability functions. Probability functions are normally assumed to be real-valued and obeying the Kolmogorov axioms: monotonicity, unit measure, and countable additivity. Thus it may appear that first-order reasoning about probabilities should require a proper *first-order* characterization of the reals which cannot be complete due to limitations of FOL. However, this is not necessary since all that is really required is arithmetic operations and reasoning about order. Also, as Abadi and Halpern have shown [1], dealing with the reals, which are a subclass of ordered number fields, makes certain logics in this family fully undecidable which would be semi-decidable, i.e. completely axiomatizable, if probability functions are allowed to take values from any algebraic number fields, not only reals. To the best of our knowledge allowing non-standard probability functions does not cause major problems apart from some facts about reals, such as $\sqrt{2} \times \sqrt{2} = 2$, not being true in all models. As such we take the Bacchus approach and allow such functions but use Halpern's terminology (Types I, II, and III) to refer to different logics in the family.

We next proceed to describing the three logics which mainly differ in the kind of probabilities they aim at representing and reasoning about. The Type I logic deals with statistical statements while Type II deals with belief statements. The Type III logic combines the features of both but does not (by default) offer any mechanism for connecting the two kinds of probabilities. We return to that issue at the end of the section.

Type I Probabilistic Logic We start with the Type I logic (or FOPL_I) which is capable of representing and reasoning about statistical knowledge with respect to FOL theories, i.e., statements like "90% of birds fly" or "age-adjusted breast cancer mortality rate for US women of all races is 24.0 per 100,000."[11] We assume a two-sorted first-order vocabulary. The first sort consists of predicates and function names of different arity Φ and a countable set $\mathcal{X}^o$ of *object variables* $x^o, y^o, \ldots$. Intuitively, these variables range over the abstract domain as in standard FOL. The second sort is composed of constants 0 and 1, the binary function names $+$ and $\times$, the binary predicate names $>$ and $=$, and a countable set $\mathcal{X}^f$ of *field variables* $x^f, y^f, \ldots$. Intuitively, these variables range over field elements, i.e. numbers.

Object terms are simply the closure of $\mathcal{X}^o$ under function applications. Formulas of FOPL_I and field terms are defined simultaneously in the following way:

- $0, 1$ and all field variables are field terms.

[11] `http://seer.cancer.gov/statfacts/html/breast.html`

- If P is an n-ary predicate name in Φ and $t_1, \ldots, t_n$ are object terms, then $P(t_1, \ldots, t_n)$ is an atomic formula.
- If t and s are field terms then so are $t + s$ and $t \times s$.
- If t and s are field terms then $t = s$ and $t > s$ are atomic formulas.[12]
- If ϕ is a formula then $w_{\vec{x}}(\phi)$ is a field term, where $\vec{x}$ is a vector of n object variables.
- Formulas of the form $w_{\vec{x}}(\phi|\psi) = t$ and $w_{\vec{x}}(\phi|\psi) > t$ are abbreviations for $w_{\vec{x}}(\phi \wedge \psi) = t \times w_{\vec{x}}(\psi)$ and $w_{\vec{x}}(\phi \wedge \psi) > t \times w_{\vec{x}}(\psi)$ respectively.
- The set of formulas is closed under conjunction, negation, and universal quantification. Both object and field variables can be bound by the universal quantifier.
- Logical symbols $\vee, \rightarrow$, and $\exists$ are standard abbreviations defined in terms of $\wedge, \neg$, and $\forall$. Field predicates $\leq$ and $\geq$ are defined in a similar way.

Following Halpern [76, 1], we define a Type I structure over Φ to be a tuple $M = (D, \pi, \mu)$ where D is a domain, π is the standard first-order interpretation function which maps predicates and functions from Φ to predicates and functions over D, and μ is a discrete probability distribution over D. μ is extended to $\mu^n : 2^{D^n} \rightarrow [0, 1]$ by defining $\mu^n(d_1, \ldots, d_n) = \mu(d_1) \times \cdots \times \mu(d_n)$ and closing under countable union. Valuation is defined as a function v which maps each object variable into an element of D and each field variable into an element of $\mathbb{R}$. Next we inductively associate each formula with a truth value by using a structure M and a valuation function v ($[t]_{(M,v)}$ stands for the element of D (resp. $\mathbb{R}$) to which the object (resp. field) term t is mapped). Using the standard convention we write $(M, v) \models \phi$ as an abbreviation of "ϕ is true in (M, v) (or (M, v) satisfies ϕ)". The definition follows the corresponding definition in FOL, so the following few clauses suffice:

- $(M, v) \models t = s$ iff $[t]_{(M,v)} = [s]_{(M,v)}$;
- $(M, v) \models \forall x^o \phi$ iff $(M, v[x^o/d]) \models \phi$ for all $d \in D$, where $v[x^o/d]$ stands for the valuation that maps x^o to d and is otherwise equivalent to v;
- $[w_{\vec{x}}(\phi)]_{(M,v)} = \mu^n(\{(d_1, \ldots, d_n) : (M, v[x_1/d_1, \ldots, x_n/d_n]) \models \phi\})$.

As usual those formulas which are true in some Type I structures are called satisfiable and those which are true in all structures are called valid. We write $M \models \phi$ if M satisfies ϕ for all valuations.

[12] If object equality is considered part of the language then $t = s$ is also an atomic formula whenever t and s are object terms.

Typically first-order formulas ϕ occurring in $w_{\vec{x}}(\phi)$ are open first-order formulas with n free object variables. In that case their interpretation can be informally read as the probability that a randomly chosen vector $(d_1, \ldots, d_n)$ of domain objects satisfies ϕ. In what follows we will ignore field terms $w_{\vec{x}}(\phi)$ where $\vec{x}$ does not correspond to the vector of free object variables in ϕ. In particular, we will ignore field terms over closed formulas because they can only be mapped to either 0 or 1 in any Type I structure (see [76] for the proof of this fact).

FOPL_I is capable of representing a broad range of statistical statements, including conditional probabilities, and various notions of independence (see the examples below).

Example 2.1 (Statistical Formulas in FOPL_I)**.**

- *Conditional probabilities:* $0.12 \leq w_x(woman\_with\_breast\_cancer(x)|woman(x)) \leq 0.13$. *Prevalence of breast cancer among women is 12%–13%.*
- *Qualitative probabilities:* $w_x(person\_with\_breast\_cancer(x)|woman(x)) \geq 10 \times w_x(person\_with\_breast\_cancer(x)|man(x))$. *Breast cancer is more than ten times more prevalent among women than men.*
- *Independence:* $w_x(person\_with\_breast\_cancer(x)|man(x) \wedge \exists y\, dog(y) \wedge loves(x,y)) = w_x(person\_with\_breast\_cancer(x)|person(x))$. *Loving dogs does not affect the risk of developing breast cancer risk.*

Type II Probabilistic Logic The Type II probabilistic logic (FOPL_II) has a very similar syntax to the one of FOPL_I. The only difference is that instead of field terms of the form $w_{\vec{x}}(\phi)$, FOPL_II allows for terms $w(\phi)$ which are informally interpreted just as "the probability of ϕ". Since the Type II logic does not deal with probability distributions over the domain, there is no notion of a random choice of $\vec{x}$ that will satisfy ϕ with some probability. Instead, the logic allows talking about the probability of (typically closed) formulas which is defined with respect to *possible worlds*.

The notion of a "possible world" is made precise in the following way. A Type II probability structure is a tuple $M = (D, S, \pi, \mu)$, where D is the domain, S is a set of possible worlds (or states), π is a world-specific first-order interpretation function (i.e. it may interpret function and predicate names differently in different worlds), and μ is a discrete probability distribution over S. The key difference between the Type I and Type II semantics is that now the probability distributions are taken over the set of worlds S and not over the domain D.

A Type II structure M, a world $s \in S$, and a valuation v collectively associate every object and field term with an element of D and $\mathbb{R}$ respectively, and every formula ϕ with a truth value. As before we write $(M, s, v) \models \phi$ if the tuple (M, s, v) maps ϕ to

true. Next we present few important clauses to define the relation $\models$ for FOPL_{II} (see [12] for the complete list):

- $(M, s, v) \models P(x)$ iff $v(x) \in \pi(s)(P)$. Note that each world can be regarded as a first-order structure with its own interpretation function. All worlds are assumed to share the same domain but this condition can be lifted [76].
- $(M, s, v) \models t_1 = t_2$ iff $[t_1]_{(M,s,v)} = [t_2]_{(M,s,v)}$;
- $(M, s, v) \models \forall x^o \phi$ iff $(M, s, v[x^o/d]) \models \phi$ for all $d \in D$;
- $[w(\phi)]_{(M,v)} = \mu(\{s \in S | (M, s, v) \models \phi\})$. Here the interpretation of field terms of the form $w(\phi)$ does not depend on a world since it is defined as a probability of all worlds in which ϕ is true.

The semantics of FOPL_{II} is generic in the sense that it allows for any (non-empty) set to be used as a set of possible worlds. However, it is common to take S as the set of all interpretations of symbols in Φ over D (see, for example, [118]). In what follows, especially Chapter 4, we refer to such choice of worlds as "natural" and omit π in the structure (since every state s is by itself an interpretation). Also, it is reasonable to require that all formulas appearing in terms $w(\phi)$ are closed (similarly to how we required them to be open in the previous section). In that case all components of M become fixed, so we will, for example, write $M \models w(\phi) \leq t$ instead of $(M, v) \models w(\phi) \leq t$.

Example 2.2 (Belief Formulas in FOPL_{II})**.**

- *Ground beliefs:* $w(loves(Mary, Fido)) \geq 0.9$*. It is believed with probability more than 90% that Mary loves Fido.*
- *Conditional and qualitative probabilities:* $w(loves(Mary, Fido)|dog(Fido)) \geq w(loves(Mary)|cat(Fido))$*. It is more likely that Mary loves Fido if it is a dog than if it is a cat.*

Type III Probabilistic Logic and Direct Inference Finally, we present the Type III logic (or FOPL_{III}) which has been designed to combine features of FOPL_{I} and FOPL_{II} [76] in order to represent and reason about different kinds of probability. The syntax of FOPL_{III} allows for both types of field terms, namely $w_{\vec{x}}(\phi)$ and $w(\phi)$, and, furthermore, it allows for their nesting. Consider the following example of a well-formed formula in FOPL_{III} (it says that the *degree of belief* that Mary has breast cancer given the available *statistics* about women with BRCA mutations is five times as high as the risk of an average woman):

$$(w(woman\_with\_breast\_cancer(Mary) \mid$$
$$0.6 \leq w_x(woman\_with\_breast\_cancer(x) \mid woman\_with\_brca\_mutation(x)) \leq 0.8)$$
$$\geq 5 \times w_x(woman\_with\_breast\_cancer(x) \mid woman(x))$$

A Type III probability structure is essentially a combination of structures of Type I and II. It is a tuple $M = (D, S, \pi, \mu_D, \mu_S)$, where D, S, π are defined exactly as for FOPL_{III}, and μ_D and μ_S are discrete probability distributions over D and S respectively. Such structure associates a truth value with formulas of both kinds $w_{\vec{x}}(\phi)$ and $w(\phi)$ as follows:

- $[w_{\vec{x}}(\phi)]_{(M,v)} = \mu^n(\{(d_1, \ldots, d_n) : (M, v[x_1/d_1, \ldots, x_n/d_n]) \models \phi\})$;
- $[w(\phi)]_{(M,v)} = \mu(\{s \in S | (M, s, v) \models \phi\})$.

While FOPL_{III} provides means of *expressing* different kinds of probabilities it does not provide any mechanism for *connecting* them. From a very cautious, purely probabilistic point of view such connections may not exist. However, in many applications, most prominently in actuarial reasoning (reasoning about risk), it is desirable that general statistical knowledge affects beliefs about specific individuals. Consider the situation in which the knowledge base contains the following statements:

I. An average US woman has a 12%–13% chance of developing breast cancer in her lifetime: $0.12 \leq w_x(woman\_with\_breast\_cancer(x) | woman(x)) \leq 0.13$.

II. Given that a US woman has mutations in the BRCA genes, her chance of developing breast cancer is between 60% and 80%:

$$0.6 \leq w_x(woman\_with\_breast\_cancer(x) | woman\_with\_brca\_mutation(x)) \leq 0.8$$

III. Mary is from the US and she does have BRCA(1) mutation: $woman(Mary)$, $w(woman\_with\_brca\_mutation(Mary))$.

IV. Mary loves dogs: $\exists y\ dog(y) \wedge loves(Mary, y)$.

It is intuitively unfortunate that the formulas I and II place no constraints on the field term $w(woman\_with\_breast\_cancer(Mary))$ even in the presence of III. This is so because the probability distributions μ_D and μ_S are effectively separated, so the statistical formulas constrain the former but not the latter. This separation can be justified because $Mary$, or any particular individual, can be an exceptional individual whose gene mutations have nothing to do with breast cancer (after all, she may have a high chance of dying from other reasons within a very short time). However, in general it is often desirable to be able to entail degrees of belief from statistical information.

This issue has a rich history and a number of mechanisms going under the generic name of *direct inference* have been proposed. Many of them are strongly related the so called *reference class* reasoning, which is first formulated by Reichenbach [167] and then substantially developed by Kyburg [123]. According to this approach computing the degree of belief in a statement about a particular object, such as Mary, involves selecting the most specific class of objects such that i) there are reliable statistics about it and ii) the object in question is a member of that class. In the example above that method would yield the probability of ≥ 0.9 for the belief statement $woman\_with\_breast\_cancer(Mary)$ because $Mary$ is a member of the class of people with BRCA gene mutations which is statistically associated with ≥ 0.9 chance of developing breast cancer.[13]

Reichenbach formulated the generic principle of direct inference but did not develop a concrete method of computation. A great deal of research has been carried out to formalize a set of desirable properties of such a method and eventually mechanize it (see esp. [122]). In particular, most approaches try to satisfy such important requirements as the capability of preferring more specific information and ignoring some irrelevant information. In the above example it seems clear that the statement II should be preferred to I when entailing the probability of $woman\_with\_breast\_cancer(Mary)$ because women with BRCA mutations is a more specific class of objects. At the same time, knowledge that Mary loves dogs should be ignored as irrelevant since nothing is known about the incidence of breast cancer among dog lovers. These considerations are intuitively appealing but there are reasonably practical situations when selecting the most specific class is problematic. We briefly outline some common issues (a more complete description can be found in [12, 15, 14]).

Preferred Super Classes The preference for specificity may fail in situations when the statistics for a subclass can be considered less reliable than for the superclass. Assume we have taken out the statement II and added two extra statements to the example above: **V.** Mary has a postgraduate degree and **VI.** Women with advanced degrees have a 2%–95% chance of developing breast cancer.[14] Now the specificity rule tells us that the statement VI should be preferred to I so the inferred risk for Mary should be 2%–95%. However, the statement VI places very weak constraints on probability distributions, possibly because only very few highly educated women have been studied. In that case one may still prefer the tighter statement I, thus trading specificity for the "quality" of the statistics. This

[13] Reasoning about a particular object can be generalized to reasoning about a collection of particular objects. In that case, the reference class becomes an extension of an n-ary predicate containing the individuals. A particularly interesting special case is that of *reference properties*. See [123] and Section 5.8 in [12] for more details.

[14] According to some research women with higher social status are at a somewhat higher risk than others but the findings are controversial [55].

intuition is captured by the strength rule, which is one of Kyburg's preference rules for reference class reasoning [123].

Disjunctive Reference Classes Disjunctions can bring about problems during reference class reasoning especially when applied to small classes (singletons, in the extreme case). For example, consider the class:

$$A(x) = \neg woman\_with\_breast\_cancer(x) \wedge woman\_with\_brca\_mutations(x) \vee spouse\_of(x, John)$$

and the extra statements $spouse\_of(Mary, John)$ and $\forall x, y, z.spouse\_of(x, y) \wedge spouse\_of(z, y) \rightarrow x = y$ (Mary is a spouse of John and two people cannot both be spouses of a single person). $A(x)$ is a subclass of $woman\_with\_brca\_mutations(x)$ so technically it should be a preferred reference class for Mary thus blocking the desirable entailment of risk. Some authors [123, 165] simply disallow disjunctive reference classes in an attempt to avoid this difficulty but such a syntactic solution can cause other problems, for example, miss implicit disjunctions or block useful classes [12, 14].

Redundant Statistics Consider the case when the user decided to add the statement $w_x(woman\_with\_breast\_cancer(x)|woman(x) \wedge \exists y.dog(y) \wedge loves(x, y)) = w_x(woman\_with\_breast\_cancer(x)|woman(x))$ to our example, perhaps trying to explicitly capture the fact that loving some dog is irrelevant to the risk of developing breast cancer. Now, since Mary is both a dog lover and a woman with BRCA mutations there two equally specific classes: $(woman(x) \wedge \exists y.dog(y) \wedge loves(x, y))$ and $(woman(x) \wedge woman\_with\_brca\_mutations(x))$. Intuitively, the second class should be preferred but there is nothing in the KB that would help to select it.

Sampling Finally, suppose that all we know is the statement I and that Mary is some woman, not necessarily an American. In certain cases it might be reasonable to assume that US women represent a fair sample of all women with respect to breast cancer. In essence this is the same kind of assumption as that dog lovers are no different from other women with respect to breast cancer but it is not supported by the basic preference based approach [15].

As you can see from the few cases outlined above direct inference is inherently related to the ability to make certain assumptions and then retract them as new knowledge becomes available or, in other words, it requires some capability of *non-monotonic reasoning* for which number of formalisms have been developed. One particularly related is the default logic developed by Reiter [168]. It explicitly separates classical formulas and defeasible formulas (or defaults), i.e., those that admit exceptions and

can be retracted. In this thesis we are most interested in the so called *statistical interpretation* of defaults which is pioneered by Bacchus [12]. We first outline the original, expectation based approach and then proceed to several other direct inference techniques. All of them can be applied to probabilistic extensions of DL and, as we argue in Chapter 4, may well be more appropriate than the method employed in P-$\mathcal{SROIQ}$.

Bacchus augments the language of FOPL_{III} with the expectation operator E. If t is a field term, e.g., a formula of the form $w_{\vec{x}(\phi)}$, then $E(t)$ is a new field term which intuitively denotes the expectation of the probability of ϕ across all possible worlds. More formally, a Type III probability structure M and a valuation v map such terms it into elements of $\mathbb{R}$ in the following way: $[E(t)]_{M,v} = \sum_{s \in S} \mu_S(s) \times [t]_{(M,v)}$. Simply put, E is a unary function that maps probability terms into numbers which are rigid(i.e., do not depend on a state). The operator has a few interesting properties proved in [12]. It allows connecting statistical probabilities and degrees of belief via the notion of randomization.

Definition 2.5 (Randomization, Bacchus [12])**.** *Let ϕ be a formula in FOPL_{I} and $\mathcal{P}$ be a set of formulas in FOPL_{I}. If $(c_1, \ldots, c_n)$ are n distinct object constants appearing in both $\mathcal{P}$ and ϕ and $(v_1, \ldots, v_n)$ are n distinct object variables not appearing in both $\mathcal{P}$ and ϕ, then $\mathcal{P}^{\boldsymbol{v}}$(resp. $\phi^{\boldsymbol{v}}$) denotes the new formula that results from replacing c_i by v_i in $\mathcal{P}$ (resp. ϕ) for every $i \in (1, \ldots, n)$.*

Bacchus' direct inference principle states that if $\mathcal{P}$ is a set of formulas in FOPL_{I} which represent all statistical knowledge for an agent then their degree of belief in a closed formula ϕ should be computed via the following equality:

$$w(\phi) = E(w_{\vec{v}}(\phi^v | \mathcal{P}^v)) \tag{2.2}$$

Informally, by replacing constants by random designators this principle says that the degree of belief in ϕ should be equal to the expected probability than a random tuple of domain objects satisfies ϕ given that it satisfies $\mathcal{P}^v$. Conditioning on the entire knowledge base is equivalent to using the most specific reference class for object constants appearing in ϕ (see examples in [12]).

Unfortunately, this principle is insufficient for dealing with the Reichenbach's original problem of "reliable statistics". For example, the probability that Mary will develop breast cancer will be equal to the term $E(w_v(woman\_with\_breast\_cancer(v) \mid woman\_with\_brca\_mutations(v) \wedge \exists y.dog(y) \wedge loves(v, y))))$. The problem is that the probability theory does not sanction inheritance of statistical properties, i.e., the constraints on risk of developing breast cancer given BRCA gene mutations do not say anything about the risk of developing breast cancer given the mutations *and* something

else. Therefore Bacchus' solution is to assume *non-monotonically* that:

$$E(w_v(woman\_with\_breast\_cancer(v) \mid woman\_with\_brca\_mutations(v) \wedge \exists y\ dog(y) \wedge loves(v,y)))) = E(w_v(woman\_with\_breast\_cancer(v) \mid woman\_with\_brca\_mutations(v))$$

This formula asserts that loving dogs is irrelevant to the risk of developing breast cancer. Such formulas need to be added for every piece of irrelevant information in order to support probabilistic inheritance.

This example illustrates the essence of Bacchus' mechanism. It deals with most of the issues outlined above by *explicitly* stating all assumptions. This has both strong and weak points. On the bright side it allows the agent to keep track of everything that has been assumed while being very cautious and probabilistically sound of what has not been assumed. Furthermore, it can be proved that all degrees of belief that can inferred through Bacchus' direct inference from a consistent statistical knowledge base are, in fact, probabilities and represent a probabilistically consistent theory [12]. On the other hand, this approach, if not complemented by some sort of relevance theory which guides the process of generating assumptions, can easily lead to an unmanageable (or even infinite) number of formulas. Also, to the best of our knowledge, its computational properties have not been sufficiently investigated and no implementation attempts have ever been made.

We now move to alternative approaches to direct inference which do not require explicit syntactic assertions but are based on semantic rules determining probability distributions over possible worlds based on statistical information. They are based on a variant of FOPL_{III} logic with the following simplifications [15]:

- The domain D in model structures is finite (but not necessarily bounded). This restriction enables the next two simplifications.
- Probability distributions over D (i.e. μ_D) are uniform. In this case terms of the form $w_{\vec{x}}(\phi)$ are interpreted as proportions of domain elements that satisfy ϕ.
- The set of states S is simply the set of all first order structures over D.

The basic idea behind the approaches dates back to the early *principles of indifference* or *insufficient reason* which state that if the agent does not have knowledge to regard one situation more preferable to another then they should be assigned equal probability [186, 108]. It is assumed that first-order statistical knowledge is all that the agent knows and it can be used to describe the set of possible situations which

should be treated as equally likely. The methods differ, however, in what they take as a "possible situation".

The first method, called *random worlds*, associates situations with worlds in $\mathrm{FOPL}_{\mathrm{III}}$ model structures. First-order statistical knowledge determines the set of worlds which are assigned equal probability. The uniform probability distribution exists because the set of first-order structures over a finite signature and a finite domain is finite. The other two methods, called *random structures* and *random propensities*, behave similarly but they group worlds based on different criteria. The first method groups worlds which are isomorphic with respect to predicates in the vocabulary. Intuitively, if sets of individuals are indistinguishable by predicates they "belong to," then they can treated as equivalent. According to the last method, each world can be specified as a tuple $(e_1, \ldots, e_k)$ where e_i is the number of domain objects satisfying the unary predicate P_k.[15] Intuitively, each situation characterizes the propensity that an individual (or a set of individuals) satisfies each predicate. The last two methods uniformly divide the probability among worlds in each equivalence class.

All three methods select a unique Type II probability model (i.e., a canonical model) which is used to infer point-valued degrees of belief. They can be understood in terms of the properties of that model. The random worlds approach selects the maximum entropy model and, therefore, is similar to probabilistic logics of maximum entropy [16, 162, 160] and objective Bayesianism [189]. The random structures method selects the model which represents the center of mass while the the random propensities method uses the model which maximizes the statistical independence among predicates in the vocabulary. All methods have some desirable properties of direct inference. Namely, they all generalize deductive reasoning and support inference using the most specific information in non-controversial cases [15, 14]. For example, all of them entail $0.6 \leq woman\_with\_breast\_cancer(Mary) \leq 0.8$ if the example above contains statements I–III. However, they differ in other properties such as support of sampling and the ability to ignore seemingly irrelevant information (for example, the random structures method does not ignore the fact that Mary loves dogs). Note that the methods do not require explicit rules for selecting the reference class and, therefore, avoid the problem of disjunctive reference classes.

A number of authors have pointed out that it is unlikely that a domain-independent direct inference method that would generate optimal degrees of beliefs in all cases is realizable. Arguments in support of this view range from philosophical points to specific examples, such as the well known Nixon diamond scenario. Therefore, it is important to understand the properties of different methods so as to be able to apply those which work best in a given situation. We believe that all the presented methods are relevant

[15] We assume for simplicity that the signature only contains unary predicates although the method makes perfect sense in the general case, see [15, 14].

for probabilistic Description Logic and can supersede the direct inference mechanism used in P-$\mathcal{SROIQ}$ (see Chapter 4 for more details).

Probabilistic Description Logics

Due to the importance of Description Logics as subsets of FOL it is no surprise that a number of authors proposed numerous ways to extend them with probabilities. Here present few formalisms which illustrate different approaches to probabilistic DLs (a more complete list can be found in, e.g., [136, 38]). Analogously to first-order languages probabilistic DLs can be classified based on whether they use graphical models as an underlying representation or a reasoning mechanism, or not (we call the latter *purely logical* probabilistic DLs).

Purely Logical Probabilistic DLs Languages in this family can generally be understood as fragments of one of first-order logics of probability presented in previous sections.[16] Early works include those of Heinsohn [82] and Jaeger [92]. The latter is especially related to P-$\mathcal{SROIQ}$ since it also has a Type II semantics for interpreting probabilities attached to TBox and ABox axioms. The major difference is the direct inference mechanism which, in case of [92] is cross-entropy minimization. Heinsohn's formalism does not support probabilistic assertions on concept and role instances.

Important recent works in this area include P-$\mathcal{SROIQ}$'s predecessor, named P-$\mathcal{SHOQ}$(D) [65], probabilistic ABoxes by Dürig and Studer [52], and subjective probabilistic DL of Lutz and Schröder [139]. P-$\mathcal{SHOQ}$(D) is syntactically very similar to P-$\mathcal{SROIQ}$ but has a domain-based probabilistic semantics (i.e., should be understood as a fragment of FOPL_{I} rather than FOPL_{II}). The $\mathcal{PALC}$ language presented in [52] is different from most other approaches to probabilistic DL because it is centered around probabilistic ABoxes. The authors developed a domain-based semantics that treats every individual and every pair of domain individuals as *independent* random variables in an attempt to "reduce search space" but did not present any algorithms or computability results.

The language of Lutz and Schröder [139], Prob-$\mathcal{ALC}$, deserves special attention as the first probabilistic DL that was designed as by choosing a fragment of FOPL (in that case, FOPL_{II}) and thus carefully defines the sort of probabilities it deals with, i.e., degrees of belief. Differently from other probabilistic DLs Prob-$\mathcal{ALC}$ provides modal-like operators for constructing probabilistic *concepts* like "all people whose probability

[16] We do not mean that any knowledge base can be translated to FOPL while preserving all entailments since the logics can include extra semantic features, such as direct inference via cross-entropy minimization as in [92]. However, an appropriate FOPL can be used as a basic framework for understanding the kind of probabilities they deal with.

of having breast cancer is over 0.9" (contrast it with, e.g., P-$\mathcal{SROIQ}$ where all concepts are normal $\mathcal{SROIQ}$ concepts which can appear in probabilistic *axioms*). It also supports probabilistic ABoxes but, in this case, by attaching probabilistic to axioms or, more generally, to sets of axioms. The authors present a family of Prob-$\mathcal{ALC}$-like languages which fall into to a range of complexity classes, from `PTime` (i.e., same as $\mathcal{EL}^{++}$) to undecidable. These formalisms are still in their infancy so practical reasoning algorithms have not been developed.

Bayesian or Markov Probabilistic DLs Less closely related are combinations of DL with Bayesian or Markov networks. The first language in this family is P-CLASSIC [119], which is a probabilistic extension of the early DL CLASSIC [20]. A knowledge base in P-CLASSIC includes *probabilistic classes* which are Bayesian networks over a class' properties, e.g., the number of role fillers. The idea is similar to Object Oriented Bayesian Networks [120] except that query answering in P-CLASSIC is done not by instantiating ground Bayesian networks (like most KBMC formalisms do) but by *lifted inference*, i.e., by reasoning at the level of predicates rather than ground terms.

Newer approaches to combining DL with graphical models include works on Bayesian DL-Lite [40], credal $\mathcal{ALC}$ [38], and Markov DL [70]. Bayesian DL-Lite [40] is geared towards *tractable* query answering in a Bayesian extension of DL-Lite [27]. It uses Bayesian networks to annotate classical DL-Lite axioms to specify uncertain events when the axiom holds. Importantly, the language allows for query answering in LogSpace in data complexity, i.e., retains the good computational properties of DL-Lite. Credal $\mathcal{ALC}$ developed by Cozman and Polastro [38] is a KBMC formalism which treats probabilistic TBoxes as relational graphical models. It has similarity with MEBN but i) is less expressive by restricting the classical component to $\mathcal{ALC}$ and ii) uses credal networks, which are generalizations of Bayesian networks. The basic sort of axioms in the language is probabilistic concept inclusion of the form $P(D|C) = \alpha$, which is very similar to P-$\mathcal{SROIQ}$ except that only concept names are allowed for conclusions. The main inference task is computing the probability of a concept assertion given the rest of KB. It is performed by using first-order variable elimination techniques to reduce the size of instantiated networks. Finally, the formalism developed in [70] is based on the DL fragment of Markov logic [49]. Similarly to P-$\mathcal{SROIQ}$, the language separates between deterministic (i.e., classical) and probabilistic axioms, where classical knowledge is used to prune out impossible worlds. Reasoning is performed by Gibbs sampling to approximate the full joint probability distribution over the instantiated Markov network. To our knowledge, no practical implementation has yet been reported and evaluated.

2.3 P-$\mathcal{SROIQ}$: A Probabilistic Description Logic

This section describes the probabilistic description logic P-$\mathcal{SROIQ}$, namely, its syntax, semantics, standard reasoning procedures, and complexity results. It also presents the reasoning algorithms as originally developed by Lukasiewicz and Giugno [65, 136]. Discussion of various syntactic and semantic properties of P-$\mathcal{SROIQ}$ is deferred to Chapter 4 while optimized reasoning procedures are presented in Chapter 5.

2.3.1 Syntax and Semantics

P-$\mathcal{SROIQ}$ [136] is a probabilistic generalization of the DL $\mathcal{SROIQ}$ [88]. It provides means for expressing probabilistic relationships between arbitrary $\mathcal{SROIQ}$ concepts and a certain class of probabilistic relationships between classes and individuals. Any $\mathcal{SROIQ}$, and thus OWL 2 DL, ontology can be used as a basis for a P-$\mathcal{SROIQ}$ ontology, which facilitates transition from classical to probabilistic ontologies.

Syntax

The main additional syntactic construct in P-$\mathcal{SROIQ}$ is the conditional constraint.

Definition 2.6 (Conditional Constraint)**.** *A conditional constraint is an expression of the form $(D|C)[l,u]$, where C and D are concept expressions in $\mathcal{SRIQ}$ (i.e., $\mathcal{SROIQ}$ without nominals) called* ***evidence*** *and* ***conclusion****, respectively, and $[l,u] \subseteq [0,1]$ is a closed real-valued interval. In the case where C is $\top$ the constraint is called* ***unconditional****.*

Ontologies in P-$\mathcal{SROIQ}$ are separated into a classical and a probabilistic part. It is assumed that the set of individual names N_I is partitioned onto two sets: classical individuals N_{CI} and probabilistic individuals N_{PI}.

Definition 2.7 (PTBox, PABox, and Probabilistic Knowledge Base)**.** *A* ***probabilistic TBox*** *(PTBox) is a pair $PT = (\mathcal{T}, \mathcal{P})$ where $\mathcal{T}$ is a classical $\mathcal{SROIQ}$ TBox and $\mathcal{P}$ is a finite set of conditional constraints. A* ***probabilistic ABox*** *(PABox) is a finite set of conditional constraints associated with a probabilistic individual $o_p \in N_{PI}$. A* ***probabilistic knowledge base*** *(or a probabilistic ontology) is a triple $PO = (\mathcal{T}, \mathcal{P}, \{\mathcal{P}_{o_p}\}_{o_p \in N_{PI}})$, where the first two components define a PTBox and the last is a a set of PABoxes.*

Informally, a PTBox constraint $(D|C)[l,u]$ expresses a conditional statement of the form "if a *randomly* chosen individual is an instance of C, the probability of it being an instance of D is in $[l,u]$". A PABox constraint, which we write as $(D|C)_o[l,u]$ where o is a probabilistic individual, states that "if a *specific* individual (that is, o) is an

instance of C, the probability of it being an instance of D is in $[l, u]$". That distinction is important for default reasoning in P-$\mathcal{SROIQ}$.

Definition 2.8 (Probabilistic Signature)**.** *Given a probabilistic knowledge base PO let $\mathcal{CE}(PO)$ be the set of all concept expressions that appear either as evidence or conclusion in some conditional concept (in the PTBox or in a PABox). Then* ***probabilistic signature*** *of PO, denoted as $\Phi(PO)$, is the smallest set of concept expressions such that i) no expression is a union, intersection, or complement of other expressions and ii) its closure under union, intersection, and complementation is a superset of $\mathcal{CE}(PO)$.*

$\Phi(PO)$ (or simply Φ when the ontology is clear from context) is a finite set because $\mathcal{CE}(PO)$ is finite. It can be computed by starting from $\Phi = \mathcal{CE}(PO)$ and exhaustively applying the following rules, where C and D are concept expressions:

- If $C \sqcup D \in \Phi$, then $\Phi \leftarrow (\Phi \cup \{C, D\}) \setminus \{C \sqcup D\}$;
- If $C \sqcap D \in \Phi$, then $\Phi \leftarrow (\Phi \cup \{C, D\}) \setminus \{C \sqcap D\}$;
- If $\neg C \in \Phi$, then $\Phi \leftarrow (\Phi \cup \{C\}) \setminus \{\neg C\}$;

The process of applying the rules will terminate since every rule reduces the syntactic length of expressions in Φ.

We conclude with an example which shows a small ontology which, first, defines breast cancer (BRC), duct cancer, and lobular cancer using DL, second, expresses generic knowledge that 10%–11% of cancer incidence among women is breast cancer, and third, states that *Mary* has $\geq 90\%$ chance of having duct cancer.[17]

Example 2.3 (Fragment of a probabilistic ontology about breast cancer)**.**

$$\begin{aligned}
\mathcal{T} = \{&BRC \equiv Cancer \sqcap \exists occursIn.\exists partOf.Breast \\
&Duct \sqsubseteq \exists partOf.Breast, Lobule \sqsubseteq \exists partOf.Breast\} \\
\mathcal{P} = \{&(Woman \sqcap \exists disease.BRC \mid Woman \sqcap \exists disease.Cancer)[0.1, 0.11]\} \\
\mathcal{P}_{Mary} = \{&(Woman \sqcap \exists hasDisease.(Cancer \sqcap occursIn.Duct))[0.9, 1]\}
\end{aligned}$$

According to Definition 2.8, $\mathcal{CE}(PO)$ and $\Phi(PO)$, where $PO = (\mathcal{T}, \mathcal{P}, \mathcal{P}_{Mary})$, are the following sets:

$$\begin{aligned}
\mathcal{CE}(PO) = \{&Woman \sqcap \exists disease.BRC, Woman \sqcap \exists disease.Cancer, \\
&Woman \sqcap \exists hasDisease.(Cancer \sqcap occursIn.Duct)\} \\
\Phi(PO) = \{&Woman, \exists disease.BRC, \exists disease.Cancer, \\
&\exists hasDisease.(Cancer \sqcap occursIn.Duct)\}
\end{aligned}$$

[17] It is often convenient to introduce new concept names, such as $WomanWithBreastCancer$, to avoid repetition of complex expressions in conditional constraints.

Semantics

The semantics of P-$\mathcal{SROIQ}$ is based on the notion of a *world*.

Definition 2.9 (World)**.** *Given the probabilistic signature Φ a **world** W is a subset of Φ. A concept $C \in \Phi$ occurs **positively**, or is **satisfied**, in a world W if $C \in W$, otherwise it is said to occur **negatively**. The satisfaction relation is extended recursively to Boolean expressions over Φ in a standard way (e.g., W satisfies $A \sqcup B$ if W satisfies A or W satisfies B).*

Finally, we extend the definition of satisfaction in a world to $\mathcal{SROIQ}$ TBoxes.

Definition 2.10 (Possible World, Index Set)**.** *A world W is a **possible world** with respect to a $\mathcal{SROIQ}$ TBox $\mathcal{T}$, written as $W \models \mathcal{T}$, if $\mathcal{T} \cup \{\{o\} \sqsubseteq C | C \in W\} \cup \{\{o\} \sqsubseteq \neg C | C \notin W, C \in \Phi\}$ is satisfiable, where o is an individual name not occurring in $\mathcal{T}$. The set of all possible worlds over Φ with respect to $\mathcal{T}$, also called the **index set**, is denoted as $\mathcal{W}_\Phi(\mathcal{T})$.* [18]

Possible worlds correspond to what is commonly known as realizable concept types in the DL literature [138]. Each world W can be thought of as a conjunctive concept expression $X \equiv (\bigsqcap_{C \in W} C) \sqcap (\bigsqcap_{C \notin W, C \in \Phi} \neg C)$ so that the world is possible iff X is satisfiable (i.e., there is a realization of the concept type given a TBox).

In what follows we assume a *linear order* of basic concepts in Φ (the ordering will become important in Section 5.1.3). Since Φ is a finite set we can denote the i-th basic concept in Φ by C_i. For a given possible world W we also use the notation W_i to denote either C_i if C_i occurs positively in W or $\neg C_i$ if it occurs negatively. For a given PTBox the order of basic concepts is fixed across all possible worlds.

Definition 2.11 (Probabilistic Interpretation, Probability of a Concept)**.** *A **probabilistic interpretation** Pr of a PTBox $(\mathcal{T}, \mathcal{P})$ is a function $Pr : \mathcal{W}_\Phi(\mathcal{T}) \to [0, 1]$ such that $\sum_{W \in \mathcal{W}_\Phi(\mathcal{T})} Pr(W) = 1$. The **probability of a concept** C, denoted as $Pr(C)$, is defined as $\sum_{W \models C} Pr(W)$. $Pr(D|C)$ is an abbreviation for $Pr(C \sqcap D)/Pr(C)$ if $Pr(C) > 0$ and undefined otherwise.*

In other words, a probabilistic interpretation is a probability distribution over possible worlds. It can be thought of as a function which maps each concept type Ct over Φ to the probability that a randomly chosen named individual is a realization of Ct.

Definition 2.12 (Satisfaction by Probabilistic Interpretation)**.** *A probabilistic interpretation Pr satisfies (is a model of) a conditional constraint $(D|C)[l, u]$, written as $Pr \models (D|C)[l, u]$, if $Pr(C) = 0$ or $Pr(D|C) \in [l, u]$. Pr satisfies (is a model of) a set*

[18] In later chapters we will often omit "w.r.t. $\mathcal{T}$" and simply write "possible world" (or $\mathcal{W}_\Phi$ instead of $\mathcal{W}_\Phi(\mathcal{T})$) when $\mathcal{T}$ is clear from context.

of conditional constraints $\mathcal{F}$ if it satisfies all constraints in $\mathcal{F}$. A PTBox $PT = (\mathcal{T}, \mathcal{P})$ is called ***satisfiable*** *if there exists a probabilistic interpretation that satisfies $\mathcal{P}$.*

Observe that a conditional constraint is satisfied by all probabilistic interpretations that assign zero probability to the evidence (we discuss the implications of this in Section 2.3.4). Given the definition of satisfaction we formulate logical consequence in a standard way.

Definition 2.13 (Logical Consequence)**.** *A conditional constraint $(D|C)[l, u]$ is a* ***logical consequence*** *of a PTBox $(\mathcal{T}, \mathcal{P})$, written as $(\mathcal{T}, \mathcal{P}) \models (D|C)[l, u]$, if all models of $(\mathcal{T}, \mathcal{P})$ also satisfy $(D|C)[l, u]$. $(D|C)[l, u]$ is a* ***tight logical consequence*** *of $(\mathcal{T}, \mathcal{P})$, written as $(\mathcal{T}, \mathcal{P}) \models_{tight} (D|C)[l, u]$ if l (resp. u) is the minimum (resp. the maximum) of $Pr(D|C)$ over all models Pr of $(\mathcal{T}, \mathcal{P})$ such that $Pr(C) > 0$.*

2.3.2 Reasoning Problems

The reasoning problems in P-$\mathcal{SROIQ}$ fall into two basic categories. The first are the standard logical problems of satisfiability and (tight) entailment. The second are the problems of default reasoning in P-$\mathcal{SROIQ}$, namely, the problem of non-monotonic lexicographic entailment and two consistency problems.

Classical Reasoning Problems

As it has been defined up to now, P-$\mathcal{SROIQ}$ can be considered as a monotonic probabilistic DL with the standard satisfiability and logical entailment problems:

Probabilistic Satisfiability (PSAT): Given a PTBox PT decide if it is satisfiable.

Tight Logical Entailment (TLogEnt : Given a PTBox PT and two $\mathcal{SROIQ}$ concepts C and D, compute rational numbers $l, u \in [0, 1]$ such that $PT \models_{tight} (D|C)[l, u]$.

Both problems are reducible to classical reasoning in $\mathcal{SROIQ}$ and Linear Programming (LP) as will be discussed in Section 2.3.3.

Default Reasoning Problems

Default reasoning in P-$\mathcal{SROIQ}$ (which is based on the lexicographic entailment by Lehmann [127]) relies on notions of default consistency. These have been adapted from the notions of consistency in propositional probabilistic default reasoning which, in turn, were originally developed in the context of default reasoning with conditional knowledge [68]. Some preliminary definitions are required to formulate the notions of consistency in P-$\mathcal{SROIQ}$ [136]:

Definition 2.14 (Verification, Falsification, Toleration)**.** *A probabilistic interpretation Pr* ***verifies*** *a conditional constraint $(D|C)[l,u]$ if $Pr(C) = 1$ and $Pr(D) \in [l,u]$. Pr* ***falsifies*** *$(D|C)[l,u]$ if $Pr(C) = 1$ and $Pr(D) \notin [l,u]$. A set of conditional constraints $\mathcal{F}$* ***tolerates*** *a conditional constraint $(D|C)[l,u]$ under a $\mathcal{SROIQ}$ TBox $\mathcal{T}$, if PTBox $(\mathcal{T},\mathcal{F})$ has a model that verifies $(D|C)[l,u]$.*

Informally, toleration means that the conditional constraint $(D|C)[l,u]$ is *not in conflict* with constraints in $\mathcal{F}$ given the classical knowledge base $\mathcal{T}$. Such notion of conflict is central to the notion of consistency which can be regarded as a possibility to resolve all conflicts in the process of default (lexicographic) reasoning.

Observe, that toleration is not symmetric in any sense of the term, in particular, that $(D|C)[l,u]$ is not tolerated by $\mathcal{F}$ under $\mathcal{T}$ does not imply that any constraint in $\mathcal{F}$ is not tolerated by $\{(D|C)[l,u]\}$ (or even $\{(D|C)[l,u]\} \cup \mathcal{F}$) under $\mathcal{T}$. Asymmetry is the key property which enables partial ordering of conditional constraints by specificity.

Definition 2.15 (z-partition, Specificity order)**.** *For a PTBox $(\mathcal{T},\mathcal{P})$ the* ***z-partition*** *is an ordered partition $(\mathcal{P}_0,\ldots,\mathcal{P}_k)$ of $\mathcal{P}$ such that each $\mathcal{P}_i$ ($i \in \{0,\ldots,k\}$) is a set of all conditional constraints from $\mathcal{P} \setminus \bigcup_{j=0}^{i-1} \mathcal{P}_j$ which are tolerated by $\mathcal{P} \setminus \bigcup_{j=0}^{i-1} \mathcal{P}_j$ under $\mathcal{T}$. The order of a conditional constraint ϕ is the order of its subset in the z-partition.*

The z-partition formalizes the specificity ordering of conditional constraints based on the notion of toleration. Conditional constraints $\mathcal{P}_i$ which are in conflict with (i.e. are not tolerated by) other constraints $\mathcal{P}_j$, but *not* vice versa, have a higher order and are considered more specific. In other words, for a given PTBox the z-partition defines a partial ordering on $\mathcal{P}$ such that conflicts can only occur between conditional constraints with different order.

Definition 2.16 (Consistency)**.** *A PTBox $PT = (\mathcal{T},\mathcal{P})$ is* ***consistent*** *if i) $\mathcal{T}$ is satisfiable, and ii) there exists the z-partition of PT. A probabilistic knowledge base $PKB = (\mathcal{T},\mathcal{P},(\mathcal{P}_o)_{o\in N_{PI}})$ is consistent if i) PTBox $(\mathcal{T},\mathcal{P})$ is consistent, and ii) $(\mathcal{T},\mathcal{P}_o)$ is satisfiable for all $o \in N_{PI}$.*

In other words, consistency of a probabilistic knowledge bases means that probabilistic knowledge about any of the individuals does not contradict the classical part $\mathcal{T}$. However, as will be shown shortly, it may (and often does) be in conflict with general probabilistic knowledge $\mathcal{P}$.

The following definition shows that the z-partition induces a specificity ordering over the set of probabilistic interpretations of $(\mathcal{T},\mathcal{P})$:

Definition 2.17 (Lexicographic preference and minimality)**.** *Given two probabilistic interpretations Pr_1, Pr_2 of a PTBox $(\mathcal{T},\mathcal{P})$ with z-partition $(\mathcal{P}_0,\ldots,\mathcal{P}_k)$, Pr_1 is lexicographically preferable (**lex-preferable**) to Pr_2 if there exists some $i \in \{0,\ldots,k\}$*

such that $|\{\phi \in \mathcal{P}_i | Pr_1 \models \phi\}| > |\{\phi \in \mathcal{P}_i | Pr_2 \models \phi\}|$ *and* $|\{\phi \in \mathcal{P}_j | Pr_1 \models \phi\}| = |\{\phi \in \mathcal{P}_j | Pr_2 \models \phi\}|$ *for all* $i < j \leq k$. *An interpretation* Pr *of* $(\mathcal{T}, \mathcal{P})$ *is lexicographically minimal (**lex-minimal**) if no other interpretation of* $(\mathcal{T}, \mathcal{P})$ *is lex-preferable to it.*

Intuitively, the lexicographic ordering relation on probabilistic interpretations is based on the idea of preferring more specific conditional constraints to less specific ones. It is called *lexicographic* because any two interpretations are compared based on how many constraints they satisfy in corresponding subsets of the z-partition in a lexicographic way (from more specific subsets to less specific). Finally, the lexicographic entailment is defined simply as a logical entailment when the set of all models of a PTBox is restricted to the lex-minimal, i.e. the most preferred, interpretations.

Definition 2.18 (Lexicographic consequence)**.** *A conditional constraint* $(D|C)[l, u]$ *is a lexicographic consequence (**lex-consequence**) of a set of conditional constraints* $\mathcal{F}$ *under a PTBox* $PT = (\mathcal{T}, \mathcal{P})$*, denoted as* $\mathcal{F} \models^{lex} (D|C)[l, u]$ *under* PT*, if* $Pr(D) \in [l, u]$ *for every lex-minimal model* Pr *of* $(\mathcal{T}, \mathcal{P})$ *that satisfies* $\mathcal{F} \cup \{(C|\top)[1, 1]\}$. $(D|C)[l, u]$ *is a tight lexicographic consequence (**tight lex-consequence**) of* $\mathcal{F}$ *under* PT*, denoted as* $\mathcal{F} \models^{lex}_{tight} (D|C)[l, u]$ *under* PT*, if* l *(resp.* u*) is the minimum (resp. the maximum) of* $Pr(D)$ *subject to all lex-minimal models* Pr *of* $(\mathcal{T}, \mathcal{P})$ *that satisfy* $\mathcal{F} \cup \{(C|\top)[1, 1]\}$.

The reasoning problems of deciding probabilistic consistency of a PTBox, probabilistic KB, and computing probability intervals under tight lexicographic entailment can now be defined in a straightforward way.

PTBox Consistency (PTCon): Given a PTBox PT decide if it is consistent.

Probabilistic Knowledge Base Consistency (PKBCon): Given a probabilistic knowledge base PKB decide if it is consistent.

Tight Lexicographic Entailment (TLexEnt): Given a PTBox $(\mathcal{T}, \mathcal{P})$ and two $\mathcal{SROIQ}$ concepts C and D, compute rational numbers $l, u \in [0, 1]$ such that $(\mathcal{T}, \mathcal{P}) \models^{lex}_{tight} (D|C)[l, u]$.

Next we show how the reasoning problems of PTCon, PKBCon, and TLexEnt can be *naively* reduced to PSAT and TLogEnt.

2.3.3 Original Algorithms

The reasoning algorithms originally developed for P-$\mathcal{SROIQ}$ were rather supposed to serve as a (constructive) proof of decidability and complexity results than be implemented in real reasoners. Their correctness is relatively obvious, just as well as their naivety. This section first explains the core PSAT and TLogEnt algorithms, and then proceeds to the consistency and TLexEnt procedures. It also explains why the algorithms are impractical in realistic scenarios.

Probabilistic Satisfiability and Logical Entailment

Following the Nilsson approach, deciding PSAT and computing TLogEnt in P-$\mathcal{SROIQ}$ can be reduced to linear programming [156]. More precisely, a PTBox $(\mathcal{T}, \mathcal{P})$ is satisfiable iff the system of linear inequalities (2.3) admits a solution.

$$\begin{aligned}
&\sum_{W \models \neg D \sqcap C} -l x_W + \sum_{W \models D \sqcap C} (1-l) x_W \geq 0, \text{ for each } (D|C)[l,u] \in \mathcal{P} \\
&\sum_{W \models \neg D \sqcap C} u x_W + \sum_{W \models D \sqcap C} (u-1) x_W \geq 0, \text{ for each } (D|C)[l,u] \in \mathcal{P} \\
&\sum_{W \in \mathcal{W}_\Phi} x_W = 1 \text{ and } x_W \geq 0, \text{ for each } W \in \mathcal{W}_\Phi
\end{aligned} \tag{2.3}$$

where each summation is over $\mathcal{W}_\Phi(\mathcal{T})$ with $\mathcal{W}_\Phi(\mathcal{T})$ being the set of all possible worlds over probabilistic signature Φ w.r.t. $\mathcal{T}$.

Analogously, the TLogEnt problem can be solved for PTBox $(\mathcal{T}, \mathcal{P})$ and concepts C, D by optimizing the objective function $\sum_{W \in \mathcal{W}_\Phi, W \models D} x_W / \sum_{W \in \mathcal{W}_\Phi, W \models D \sqcap C} x_W$ subject to the linear inequalities (2.3). Following a standard technique by Charnes and Cooper this optimization problem can be easily reduced to linear programming [30].

Proofs of correctness are straightforward [136]. Every solution to (2.3) represents a probability distribution over the set of possible worlds $\mathcal{W}_\Phi$ (i.e., a probabilistic interpretation of $(\mathcal{T}, \mathcal{P})$). Each conditional constraint in $\mathcal{P}$ is represented as two linear inequalities which are satisfied by an assignment of values to variables x_I iff the constraint is satisfied by the probabilistic interpretation. Thus, the system admits a solution iff all constraints in the PTBox are satisfied. Correctness of TLogEnt is equally straightforward.

The reduction to linear programming directly implies the decidability of PSAT and TLogEnt while complexity results also follow quite straightforwardly. In particular, both PSAT and TLogEnt in P-$\mathcal{SROIQ}$ are `N2ExpTime`-hard since the satisfiability problem in $\mathcal{SROIQ}$, which is `N2ExpTime`-complete [106], can be easily reduced to them. They are also `N2ExpTime`-complete because of the small-model property of linear programming problems (see, for example, [132]). Rigorous proofs can be found in [136].

Any straightforward implementation of PSAT and TLogEnt algorithms as presented above is going to be impractical due to the size of the linear system (2.3) (one example of such implementation is described in [151]). The number of variables in (2.3) is equal to the number of all possible worlds of $\mathcal{T}$ over $\mathcal{W}_\Phi$, which is exponential in the size of probabilistic signature Φ. Even with clever rewriting techniques, which seek to shrink the size of (2.3) [132], explicit representation of such linear systems does not appear feasible for probabilistic KBs with more than 15-20 conditional constraints.

Probabilistic Consistency

The original PTBox consistency algorithm (Algorithm 1) attempts to build the z-partition of a given PTBox. It returns either the z-partition (in which case the PTBox is consistent) or *null* (in which case it is inconsistent).

Input: PTBox $PT = (\mathcal{T}, \mathcal{P})$
Output: The z-partition $(\mathcal{P}_0, \ldots, \mathcal{P}_k)$ of $\mathcal{P}$ if it is consistent or *null* otherwise
1 **if** $\mathcal{T}$ *is unsatisfiable* **then return** *null*
2 **if** $\mathcal{P} = \emptyset$ **then return** $\emptyset$
3 $H \leftarrow \mathcal{P}$, $i \leftarrow -1$
4 **repeat**
5 $\quad i \leftarrow i + 1$
6 $\quad \mathcal{P}_i \leftarrow \{\phi \in H | \phi$ is tolerated by H under $\mathcal{T}\}$
7 $\quad H \leftarrow H \setminus \mathcal{P}_i$
8 **until** $H = \emptyset$ *or* $\mathcal{P}_i = \emptyset$;
9 **if** $H = \emptyset$ **then return** $\{\mathcal{P}_0, \ldots, \mathcal{P}_i\}$ **else return** *null*

Algorithm 1: The original PTBox consistency algorithm as presented in [136]

Algorithm 1 uses the PSAT algorithm as a subprocedure (line 6). In general, it requires $O(|\mathcal{P}|^2)$ PSAT tests (see the formal proof in [136]). Therefore, its complexity is also `N2ExpTime`, same as for PSAT (`N2ExpTime`-completeness of the PTCon follows trivially from this fact). Its optimized version is presented in Section 5.3.

The PKBCon algorithm is straightforward. It is composed of two steps: deciding PTBox consistency (by calling Algorithm 1) and a series of $|N_{PI}|$ additional PSAT tests to check that $(\mathcal{T}, \mathcal{P}_o)$ is satisfiable for every probabilistic individual $o \in N_{PI}$.

Tight Lexicographic Entailment

The idea of the original TLexEnt algorithm is based on the following notion of lexicographically minimal subsets of $\mathcal{P}$.

Definition 2.19 (Lexicographically minimal sets)**.** *Given a consistent PTBox $(\mathcal{T}, \mathcal{P})$ with a z-partition $\mathcal{P}_0, \ldots, \mathcal{P}_k$ and a set of conditional constraints $\mathcal{F}$, a subset $Q \subseteq \mathcal{P}$ is lexicographically preferable (**lex-preferable**) to a subset Q' w.r.t. $\mathcal{F}$ if for some $i \in \{0, \ldots, k\}$, $|Q \cap \mathcal{F} \cap \mathcal{P}_i| > |Q' \cap \mathcal{F} \cap \mathcal{P}_i|$ and $|Q \cap \mathcal{F} \cap \mathcal{P}_j| = |Q' \cap \mathcal{F} \cap \mathcal{P}_j|$ for all $i < j \leq k$. A subset Q is lexicographically minimal (**lex-minimal**) if no other subset of $\mathcal{P}$ is lex-preferable to it.*

Theorem 2.1 formalizes the obvious correspondence between the definitions of lex-minimal models and lex-minimal subsets of $\mathcal{P}$ (the proof can be found in [136]).

Theorem 2.1. *Given a consistent PTBox $PT = (\mathcal{T}, \mathcal{P})$, a set of conditional constraints $\mathcal{F}$, and concepts C, D, $\mathcal{F} \models^{lex}_{tight} [D|C][l, u]$ under PT iff $[l, u]$ is the union of all intervals*

$\{[l', u'] | (\mathcal{T}, \mathcal{Q} \cup \mathcal{F} \cup \{(C|\top)[1,1]\}) \models_{tight} [l', u']\}$ *where* $\mathcal{Q}$ *is the set of all lex-minimal subsets of* $\mathcal{P}$ *w.r.t.* $\mathcal{F} \cup \{(C|\top)[1,1]\}$.

Algorithm 2 uses Theorem 2.1 by first computing the set of lex-minimal subsets of $\mathcal{P}$ and then computing the tight logical entailments from those subsets. It solves $O(2^{|\mathcal{P}|})$ instances of PSAT to compute the lex-minimal subsets, therefore its complexity is also `N2ExpTime`. The TLexEnt problem consequently belongs to the `N2ExpTime`-complete complexity class.

Input: PTBox $PT = (\mathcal{T}, \mathcal{P})$, finite set of conditional constraints $\mathcal{F}$, concepts C, D
Output: $(l, u) \in [0,1]^2$ such that $\mathcal{F} \models^{lex}_{tight} (D|C)[l, u]$ under PT

if PT *is inconsistent* **then return** $(1, 0)$
else $(P_0, \ldots, P_k) \leftarrow$ z-partition of PT
$R \leftarrow \mathcal{F} \cup \{(C|\top)[1,1]\}$
if $(\mathcal{T}, R)$ *is unsatisfiable* **then return** $(1, 0)$
if $(\mathcal{T}, \mathcal{P} \cup R)$ *is satisfiable* **then** $(p, q) \leftarrow (0, 0)$
else $(p, q) \leftarrow (1, k + 1)$
while $p < q$ **do**
  $l \leftarrow \lfloor (p + q)/2 \rfloor$
  if $(\mathcal{T}, R \cup P_l \cup \cdots \cup P_k)$ *is satisfiable* **then** $q \leftarrow l$ **else** $p \leftarrow l + 1$
end
$K \leftarrow \{P_p \cup \cdots \cup P_k\}$
for $j \leftarrow p - 1$ **to** 0 **do**
  $(m, n) \leftarrow (0, |P_j|)$
  while $m < n$ **do**
    $l \leftarrow \lceil (m + n)/2 \rceil$
    $K' \leftarrow \{G \cup H | G \subseteq P_j, |G| = l, H \in K, (\mathcal{T}, R \cup G \cup H)$ is satisfiable$\}$
    if $K' \neq \emptyset$ **then** $m \leftarrow l$ **else** $n \leftarrow l - 1$
  end
  $K \leftarrow \{G \cup H | G \subseteq P_j, |G| = m, H \in K, (\mathcal{T}, R \cup G \cup H)$ is satisfiable$\}$
end
$(l, u) \leftarrow (1, 0)$
for $H \in K$ **do**
  Compute $c, d \in [0,1]$ s.t. $(\mathcal{T}, R \cup H) \models_{tight} (D|\top)[c, d]$
  $(l, u) \leftarrow (min(c, d), max(c, d))$
end
return $[l, u]$

Algorithm 2: Original TLexEnt algorithm as presented in [136]

The main problem with Algorithm 2 is that performs basically an exhaustive search over the powerset of $\mathcal{P}$ in order to compute lex-minimal models. This may well lead to an unacceptably large number of PSAT instances to be solved. Section 5.4 will present a more goal directed approach which pinpoints conflicts between subsets of conditional

constraints in order to quickly compute lex-minimal models.

2.3.4 Note on Probabilistic Coherence

One interesting feature of the P-$\mathcal{SROIQ}$ semantics is a weakness of probabilistic satisfiability in the following sense: according to the definition in Section 2.3.1 any probabilistic interpretation which assigns zero to a concept C automatically satisfies all conditional constraints with C as the evidence concept. We call such satisfiability *vacuous*. This way of resolving the well known difficulty of dealing with conditioning on zero probability does, however, lead to problems in practical applications. The following example shows one of minimal conflicts in CADIAG-2 (see Section 3.2), however the PTBox is vacuously satisfiable.

Example 2.4 (Vacuous Satisfiability)**.**

$$PT = (\{D_1 \sqsubseteq \neg D_2\}, \{(D_1 \mid C)[0.9, 0.9], (D_2 \mid C)[0.9, 0.9]\})$$

D_1 and D_2 are disjoint but PT entails that the probability of $(D_1 \sqcap D_2)$ is at least 0.8. Therefore, PT can only be satisfied by interpretations assigning zero probability to C, thus "hiding" the conflict.

The kind of semantics in which a probabilistic interpretation Pr satisfies a conditional constraint $(D|C)[l, u]$ if $Pr(C) > 0$ and $Pr(D|C) \in [l, u]$ has better theoretical properties. Unfortunately solving the PSAT problem under such semantics would be far more complicated mostly because it would require encoding the conditions $Pr(C) > 0$ for every evidence concept as strict inequalities in the linear system 2.3. Such systems cannot be solved using modern numerical methods, e.g. the simplex or interior point algorithms. A number of ways of dealing with this issue have been proposed in the literature, for example, by solving a sequence of standard linear systems which approximate the system with strict inequalities [187]. However, to the best of our knowledge there have never been a successful implementation of such approaches for probabilistic logics.

We do not aim at addressing this weakness in this thesis but believe it is important to distinguish between PTBoxes in which some conditional constraints can only be satisfied vacuously and all other satisfiable PTBoxes. The following definition is provided for such a distinction:

Definition 2.20 (Probabilistic Incoherence)**.** *Given a satisfiable PTBox $(\mathcal{T}, \mathcal{P})$, let $\mathcal{R}$ be the set of all its probabilistic models. The PTBox is called* ***probabilistically incoherent*** *if for some conditional constraint $(D|C)[l, u]$ it is true that $Pr(C) = 0$ for every $Pr \in \mathcal{R}$.*

It immediately follows from Definition 2.20 that a satisfiable PTBox PT is incoherent if and only if $PT \models_{tight} (C|\top)[0,0]$ for some evidence concept C. Such concepts are called probabilistically unsatisfiable in the sequel.[19]

Although an implementation of a sound and complete probabilistic coherence algorithm is problematic, some approximations are straightforward. We use an approximate procedure based on the following definition:

Definition 2.21 (Approximate Probabilistic Coherence)**.** *A satisfiable PTBox $(\mathcal{T}, \mathcal{P})$ is **l-coherent** if for every evidence concept C there exists a probabilistic model Pr such that $Pr(C) \geq l$.*

The following lemma, which provides a connection between l-coherence and PSAT, is an immediate consequence of this definition:

Lemma 2.1. *A PTBox $PT = (\mathcal{T}, \mathcal{P})$ is l-coherent iff the augmented PTBox $PT^l = (\mathcal{T}, \mathcal{P} \cup \{(C|\top)[l,1] | (D|C) \in \mathcal{P}\})$ is satisfiable.*

This lemma yields a sound but incomplete algorithm for testing probabilistic coherence via a straightforward reduction to PSAT. Section 3.2 demonstrate the utility of such algorithm which was the principal tool for finding all inconsistencies in the medical expert system CADIAG-2. Its main weakness is obviously the choice of l which can be difficult to make in certain scenarios. We expect it to be domain specific, for example, the requirement that no symptom should have a probability below 0.01 was a part of CADIAG-2 design. In the absence of such domain specific information one may solve a sequence of PSAT instances by successively reducing l to zero until either incoherence is proved or the PTBox is considered to be *sufficiently* coherent.

2.4 Related Work

This section describes previous work focused on developing practical reasoning procedures for the most closely related probabilistic logics and their evaluation. We first describe a number of approaches to solving large-scale *propositional* PSAT and TLogEnt all of which are similar in spirit to our PSAT algorithm. After that we proceed to proof-of-concept implementations of non-monotonic reasoning algorithms, in particular TLexEnt, for propositional probabilistic logic (the NMPROBLOG system [134]) with default constraints and P-$\mathcal{SHIQ}$(D) (the ContraBovemRufum system [151]). We do not aim at covering implementations of all probabilistic formalisms mentioned in Section 2.2, especially from the KBMC family.

[19] There is an obvious correspondence between satisfiable but probabilistically incoherent PTBoxes and satisfiable TBoxes with unsatisfiable concepts in classical description logics. Such TBoxes are sometimes called incoherent which motivated the name in the probabilistic case.

2.4.1 Propositional PSAT Solvers

After Nilsson presented his basic propositional probabilistic logic (see Section 2.2.2 and [156]) it became immediately clear that a straightforward approach to solving PSAT based on solving the system of linear inequalities (2.1) is intractable not just in the worst case but also in most practically relevant cases, i.e., where the number of probabilistic statements exceeds a few dozen. Consequently, a number of authors proposed a range of approaches to developing a reasoning procedure that would allow solving PSAT instances with hundreds of statements. The approaches can be split onto two major categories: global methods and local methods.

Global Probabilistic Reasoning

The global approach to PSAT and TLogEnt, which we also follow, is based on determining consistency of the system (2.1) or optimizing a linear function subject to the same system. It has several advantages, in particular, *completeness*, i.e., it is complete for PSAT and produces the tightest possible probability intervals for TLogEnt. It also allows using the theory of *sensitivity analysis* in linear programming to determine how the probability intervals will change in response to certain changes in probabilistic formulas. The main problems are difficulties with producing justifications for the results of PSAT and TLogEnt[20] and, most importantly, the size of the linear system (2.1).

After Nilsson's original suggestion [156] virtually all subsequent research, including ours, has revolved around *column generation* (see Section 5.1.1 and Chapter 26 in [32]). The early attempts to use that technique include works of Georgakopoulos, Kavvadias and Papadimitriou [62, 105], Hooker [83], and Jaumard, Hansen and de Aragão [98, 99, 80]. They all use the standard simplex procedure to solve partially constructed instances of (2.1) (*master* problems) but differ in their methods of solving the auxiliary optimization problem (5.4) to generate new columns (variables). Kavvadias and Papadimitriou [105] and Jaumard et al. [98, 99] generate columns by optimizing a non-linear unconstrained function over binary variables. Both methods use heuristics, namely *variable depth local search* and a combination of steepest ascent mildest descent and tabu search. However, the former method is purely heuristic and *approximate* since it can fail to find a variable, which can improve the system (2.1), even if one exists. The latter method invokes an exact algorithm for pseudo-boolean programming when the heuristics fail, so it is complete for PSAT and TLogEnt. Both methods have been evaluated and showed their ability to handle PSAT instances of 100—300

[20] Our work on analysis of probabilistic knowledge bases, in particular, pinpointing minimal unsatisfiable conflicts (see Section 5.2.1), can be regarded as a way to *explain* results of reasoning, e.g., PTBox unsatisfiability.

probabilistic formulas. Finally, Hooker's algorithm [83] is also exact and closest to ours since it generates column by solving a *constrained* integer program.

More recent works on PSAT via global column generation not just use advanced heuristics to generate columns but also consider a range of extensions to the basic PSAT formulation. Hansen et al. [79] consider imprecise probabilities which include intervals and qualitative probabilistic constraints introduced by Coletti, i.e., formulas of the form $P(A) \leq (\text{or } \geq) P(B)$ [34]. Ognjanovic et al. [158, 159, 101] deal with so called *weight formulas* which are probabilistic formulas of the form $a_1 w(\alpha_1) + \cdots + a_n w(\alpha_n) \leq (\text{or } \geq) b$, where $\{a_i\}$ and b are rational numbers, $\{\alpha_i\}$ are propositional formulas, and $w(\alpha_i)$ stands for "probability of α_i". They employ a highly efficient *variable neighborhood search* (VNS) technique [101], genetic algorithms [158], and their combination [159] for generating columns. These methods where the first to scale to 1000 probabilistic formulas [101]. Finally, we mention the recent work of de Souza Andrade et al. [45] who proposed yet another approach to producing columns by linearizing, but differently from [83], the column generation model (5.4). To our knowledge, they were the first to evaluate a PSAT algorithm on approximate translations of Bayesian networks but with very limited size (up to 50 formulas). We present the same kind of evaluation but on much larger networks in Section 6.4.

Local Probabilistic Reasoning

Local methods, as opposed to the global ones, do not attempt to solve the linear system (2.1). Instead, they use a collection of propagation rules which continuously tighten the probability bounds on formulas in question. The main advantages are that, first, the rule-based algorithms generally have the "anytime" property, i.e., they can be terminated any time to get an approximate answer to TLogEnt, second, they are computationally tractable because each rule only applies to a local set of formulas, and third, any obtained result can be easily justified by keeping track of rule applications. Unfortunately, their principal disadvantage is their inherent incompleteness: not only has it been proved that *any* finite set of rules may fail to prove unsatisfiability but it has also been shown that the approximations of real TLogEnt intervals can be arbitrarily poor [131].

The first local anytime probabilistic deduction method was presented by Frisch and Haddawy who use a handcrafted collection of rules [58]. It was later observed that such rules can be automatically synthesized by solving small PSAT instances. That idea was implemented in later deduction algorithms, namely TURBOSAT [100] and AD-PSAT [100]. These algorithms usually use only very small portions of large PSAT or TLogEnt instances to find either exact or a good approximate solution, which explains their good performance. According to [81], it takes AD-PSAT less than 2 seconds to

solve TLogEnt of more than 3000 formulas over 1000 propositional variables.

Finally, recent work by Hansen and Perron showed that the local and global approaches can be combined [81]. Their PSAT/TLogEnt algorithm uses local rules for two reasons: first, to check if the rules can detect unsatisfiability (in which case the algorithm terminates), and second, to reduce the search space for generating optimal columns. According to their results, the local method (AD-PSAT) improved the average performance by approximately 30% on satisfiable knowledge bases, which, in addition to other optimization techniques such as stabilization, enabled solving instances of up to 800 formulas.

2.4.2 NMPROBLOG and ContraBovemRufum

Lastly, we briefly present two proof-of-concept implementations of propositional probabilistic logic with default formulas and P-$\mathcal{SHIQ}$(D). The former system, named NMPROBLOG [134], was the first implementation of non-monotonic reasoning algorithms for PSAT, TLogEnt, PTCon, and TLexEnt. Although, it was built for propositional logics, the implemented algorithms are largely equivalent to the original algorithms for P-$\mathcal{SROIQ}$ described in Section 2.3.3. Interestingly, the system implements the so called *variable-strength inheritance mechanism*, which allows for controlling inheritance of probabilistic properties down the hierarchy of propositional atoms (or concept hierarchy in case of P-$\mathcal{SROIQ}$).

The second system, named ContraBovemRufum [151], is also directly relevant to our work since it is an implementation of essentially the same logic, P-$\mathcal{SHIQ}$(D).[21] The system implements the original algorithms for solving PSAT, TLogEnt, PTCon, and TLexEnt, and includes some of the initial optimizations already proposed in [136]. ContraBovemRufum was later used for a small-scale probabilistic validation of probabilistic ontology alignments [29].

Both these systems eagerly construct the linear system (2.1) in order to solve PSAT or TLogEnt. As such, their scalability is limited to one or two dozen probabilistic formulas. Nonetheless, they were interesting tools to experiment with the algorithms and features of the logics (see, for example, the discussion of the entailment behavior in [151] and in Section 4.1.2).

[21] An implementation of P-$\mathcal{SHIQ}$(D) can be upgraded to P-$\mathcal{SROIQ}$ simply by upgrading to a DL reasoner that supports $\mathcal{SROIQ}$.

Chapter 3

Applications and Case Studies

This chapter presents our studies of applicability of P-$\mathcal{SROIQ}$ and the probabilistic reasoner Pronto. First, Section 3.1 introduces the Breast Cancer Risk Assessment problem and shows what kind of modeling and reasoning can be accomplished using P-$\mathcal{SROIQ}$ (it also comments on challenges of doing so). Section 3.2 describes the application of P-$\mathcal{SROIQ}$ to an automated analysis of the knowledge base which lies in the heart of the CADIAG-2 medical expert system. The section also presents the results of analysis which has revealed numerous inconsistencies in the CADIAG-2 KB. Finally, Section 3.3 explains how P-$\mathcal{SROIQ}$ can be applied to reason about uncertain ontology alignments computed by wide diversity of presently available tools.

3.1 Breast Cancer Risk Assessment Problem

The Breast Cancer Risk Assessment (BCRA) Problem is a problem of calculating the risk of developing breast cancer (generally any of cancers originating from breast tissue) given personal data about a given individual. While breast cancer can also be developed by men, women are at a significantly higher risk,[1] so prediction, diagnosis and treatment of the disease in females have been subjects of more extensive research.

Similarly to other forms of cancer, breast cancer is associated with a number of environmental and hereditary risk factors (or exposures). Dozens of epidemiological risk factors have been identified by numerous studies, however, not for all of them the studies have produced consistent results. Those factors, such as age, which are associated with certain changes in breast cancer risk by a preponderance of studies are called *established* while other are controversial or unconfirmed, for example, having an abortion. Furthermore, the causes of an individual breast cancer often remain unknown even after it has been diagnosed. Therefore, an accurate assessment of breast cancer risk remains an open, important and challenging problem.

[1] Approximately 1 vs. 128 new cases per year in the United States as of 2008.

BCRA does look like a relevant use case for probabilistic modeling using P-$\mathcal{SROIQ}$. First, there is a great deal of background knowledge about cancer that is captured in logic-based ontologies, most prominently in the National Cancer Institute (NCI) Thesaurus [67]. Therefore it is natural to attempt to reuse it for the BCRA problem instead of duplicating it. Second, the BCRA problem is inherently uncertain because i) a large share of the available background knowledge is statistical and ii) that statistical knowledge has to be taken into account by applications, e.g. risk predictors or diagnosis tools. Finally, there is a clear lack of integration between classical background knowledge represented in ontologies and statistical knowledge presented in research papers and captured in statistical models. P-$\mathcal{SROIQ}$ can address that issue by allowing ontology engineers to "connect" medical terms in an ontology by statistical relationships while leaving the rest of background knowledge intact. Intuitively this is a more appealing option that reconstructing some background knowledge about cancer in, for example, the structure of a Bayesian network which was primarily built to capture statistics. Typical examples of such reconstruction can be found, for example, in the MUNIN system based on a causal probabilistic network [6]. The system defines terms from anatomy instead of taking advantage from rich anatomical models already formalized in OWL.[2]

In what follows we examine both advantages and the challenges of using P-$\mathcal{SROIQ}$ for this sort of modeling problems.

3.1.1 Risk Types and Assessment

The term "risk of breast cancer" is somewhat ambiguous and its meaning is context dependent. We first briefly describe various types of risk and the general methodology of calculating the so called "absolute risk", which can easily be understood by a patient.

Absolute Risk Absolute risk is the risk that a given individual will develop some form of breast cancer within a period of time. This is the kind of risk that is normally expected to be calculated by a risk assessment tool. It is typically expressed as a single number, e.g. 10%.

Relative Risk Many risk models operate with so called relative risk which is the risk of developing the disease that is *relative* to a particular combination of risk factors for an individual. For example, the statement that "a woman who never had children is at 3x increased risk of breast cancer" is a relative risk statement. More precisely, RR is the ratio of a probability that a random woman with specific exposure will develop

[2] See, e.g., the Foundational Model of Anatomy at `http://sig.biostr.washington.edu/projects/fm/AboutFM.html`.

breast cancer over the probability that a random woman without that exposure will develop breast cancer.

Relative risk for various risk factors is typically estimated by performing statistical analysis of clinical test data for exposed and unexposed groups of people. Statistical models combining different risk factors to calculate the total relative risk are called *relative risk models*. However, such models are insufficient to calculate absolute risk for an individual which also depends on factors that are external to the model.

Attributable and Baseline Risk One piece of information required to calculate absolute risk from relative risk is called *attributable risk*. It is the difference in incidence rates of breast cancer in groups of people with and without risk factors captured in the relative risk model. Intuitively it represents coverage of the model by showing how much of breast cancer is *attributed* to the risk factors that the model combines. Attributable risk enables estimation of baseline risk (or baseline hazard) which is the absolute risk of developing breast cancer for those individuals who do not have *any* of the risk factors captured in the relative risk model.

Risk Methodology While the ultimate task for BCRA tools is to calculate the absolute risk for a specific individual, to do this they have to deal with all the risk types mentioned above. The classical methodology for absolute risk assessment, which is used, for example, in the Gail model [60], includes the following steps:

1. Selecting established risk factors to be included in the assessment procedure,
2. Accumulating statistics for exposed and unexposed groups of people,
3. Developing a relative risk model which combines the chosen risk factors,
4. Estimating the attributable risk of the model and the baseline hazard of the disease,
5. Projecting individual probabilities of developing breast cancer.

In the first step model developers select risk factors i) which are strongly suspected to be associated with breast cancer and ii) for which they can gather reliable statistics by observing exposed and unexposed groups of people.

Statistical data are often accumulated by performing cohort studies, controlled experiments, or case-control studies. For instance, in case-control studies individuals are split into *cases* (those who developed breast cancer over the period of study) and controls (those who did not develop the cancer) in order to analyze differences in their risk factors. Information about risk factors can be collected by interviewing or examining the subjects.

Once the statistics has been gathered the next step is to analyze it and develop a model of the relationships between risk factors and their combinations and the risk of breast cancer. This is often done by a multivariate regression analysis in order to determine a formula that will describe how breast cancer risk will change in response to changes in individual exposures as compared to unexposed people.

The next step is to estimate coverage of the model by calculating its attributable risk. This can be done on the basis of the same case data that was used to derive the relative risk model. Assuming m risk groups, the attributable risk of AR is equal to $\sum_{i=1}^{m} \frac{\rho_i}{r_i}$ where ρ_i is the proportion of cases that are in the group i and r_i is the relative risk of the group i. The baseline hazard is simply the total observed incidence rate of breast cancer times $1 - AR$.

Finally, absolute risks for specific individuals having personal risk factors can be projected within a period of time (or even over their lifetimes) by using the standard theory of competing risks. This requires knowledge of their combined relative risk (computed by the relative risk model), the baseline hazard, and the risk that they will die from other reasons. The latter is called *mortality rate* and it is often assumed that it is constant across subjects of the study.

3.1.2 Existing Tools and Models

Before proceeding to discussion of what P-$\mathcal{SROIQ}$ can offer in the BCRA domain we give a brief overview of an increasingly popular online breast cancer risk calculator and an underlying statistical model. The calculator[3] was developed in the National Cancer Institute. It is essentially a simple Web interface to an implementation of the Gail model [60]. The interface consists of an online questionnaire which collects information about personal risk factors and passes it to the model.

The Gail model [60] is a method of assessing the risk that a woman will develop breast cancer within the next 10, 20 or 30 years.[4] It follows the risk assessment methodology described in the previous paragraph.

The relative risk model of the Gail model combines five risk factors: age at menarche (AGEMEN), age at first live birth (AGEFLB), number of previous biopsies (NBIOPS), and number of first degree relatives with breast cancer (NUMREL). In addition the model splits all women onto age categories (AGECAT) in order to allow for different proportional risk models for women under 50 and over 50. It has been derived from case-control data by using a logistic regression and has the following compact form:

[3] http://www.cancer.gov/bcrisktool

[4] We consider the original Gail model in this thesis. It has later been refined and extended in some directions, for example, to provide more accurate results for African American women. A more detailed online assessment tool has also been built, see http://halls.md/breast/risk.htm.

$$
\begin{aligned}
&- 0.74948 + 0.09401(AGEMEN) + 0.52926(NBIOPS) \\
&+ 0.21863(AGEFLB) + 0.95830(NUMREL) \\
&+ 0.0108(AGECAT) - 0.28804(NBIOPS \times AGECAT) \\
&- 0.19081(AGEFLB \times NUMREL)
\end{aligned} \tag{3.1}
$$

The coefficients from this equation are converted into relative risk coefficients so the combined relative risk for a woman can be calculated simply by multiplying her risk factors. The model estimates the baseline hazard and projects individualized absolute risk as explained in the previous paragraph except that both risks are age specific. It also assumes the risk of dying from other causes is the same for all subjects in the same age group.

Apart from risk *prognosis* models there have been developed models for *diagnosing* breast cancer. Some of them use reasoning under uncertainty methods, for instance, the Bayesian network based approach has been proposed for mammographic diagnosis of breast cancer [102]. In this study we focus on risk assessment, so diagnosis systems are considered less relevant.

3.1.3 The BCRA Problem and P-$\mathcal{SROIQ}$

This section discusses potential benefits of using P-$\mathcal{SROIQ}$ and probabilistic ontologies in general for the breast cancer risk assessment problem. First of all, it appears clear that it is unreasonable to try to build a BCRA ontology in P-$\mathcal{SROIQ}$ which would fully subsume the existing statistical models, such as the Gail model. Such models perform calculations which are easily understood in terms of standard statistical methodologies, for instance, absolute risk projections based on the theory of competing risks. Trying to perform such calculations by doing logical reasoning, even if possible, is likely to cause confusion, without bringing about any substantial benefits. Therefore we consider some alternative roles that a BCRA ontology can play in the breast cancer domain.

Relative Risk Model vs. Theory of Breast Cancer

More precisely, the following two options can be considered:

- Developing a relative risk model of breast cancer as a P-$\mathcal{SROIQ}$ ontology.
- Using a P-$\mathcal{SROIQ}$ ontology as a formal background theory of breast cancer (or a fragment thereof) which can support various high-level intelligent services, including risk assessment.

We will argue that the second option is more appropriate and feasible but let us start by considering both.

P-$\mathcal{SROIQ}$ Ontology as a Relative Risk Model We have done a preliminary investigation of constructing a BCRA ontology which represents relative risk of breast cancer associated with various exposures [110]. The classical part of the ontology provides the necessary OWL vocabulary by defining concepts describing different risk factors, kinds of breast risk, and various categories of women. The probabilistic part represents the domain statistics regarding the risk associated with risk factors and their combinations as well as some statistical connections between risk factors. The ontology contains approximately 25 risk factors and 50 conditional constraints. Lexicographic entailment was discussed as the principal mechanism for assessing the risk, so that the ontology can be seen as an attempt of building a relative risk model (option one above).

A few important lessons have been learned from that investigation. Unsurprisingly, description logic is a very useful tool for describing important breast cancer concepts in a formal and unambiguous manner. It was fairly straightforward to describe all major risk factors [178]. Second, conditional probabilistic statements accurately capture the meaning of relative risk statements. For example, the semantics of the constraint `(WomanAtHighRisk|WomanBRCAMutation[0.9,1])` agrees with the statistical nature of this statement: "in more than 90% of cases involving the BRCA (1 or 2) gene mutation the woman has developed breast cancer".[5] Third, the ontology offers a more transparent and declarative model than statistical models, for instance, the entailment results can be connected back to the statements which, in turn, can be annotated with pointers to the corresponding medical studies.

However, the issues are also substantial. The main challenge lies in using a probabilistically cautious notion of entailment for calculating total risk for combinations of risk factors. A set of women having risk factors R_1 through R_n is modeled as the intersection of sets of women having at least one of the factors. However, the probability function is not compositional, i.e., the probability of intersection (or union) is not a function of probabilities of individual sets. Therefore, the modeler is forced to either specify the risk for all necessary combinations or assume independence between certain risk factors or assume that the combined risk is proportional to the risk associated with individual factors. The first approach is infeasible due to the exponential number of combinations, the second requires support of independence assertions (which

[5] Here we claim that a proper statistical semantics for a probabilistic DL agrees with the statement, not that P-$\mathcal{SROIQ}$ semantics does so. There are some technical peculiarities in how P-$\mathcal{SROIQ}$ handles statistics (see discussion Section 4.3) but they appear to be problematic only from the philosophical, and not practical, point of view.

P-$\mathcal{SROIQ}$ lacks), and the third, which is adopted in the Gail model, is also not expressible in P-$\mathcal{SROIQ}$ since it requires non-linear constraints on probabilistic models. Moreover, even if a probabilistic logic which supports independence and other types of non-linear assertions was available, it is unlikely that its encoding of the risk model would be superior to the compact equation used in the Gail model (see 3.1).

There are also other, albeit less critical, difficulties. As discussed in Section 4.3, P-$\mathcal{SROIQ}$ does not provide a strong support of probabilistic relational structures which means that it is not possible, for example, to capture uncertainty that a woman is related to another woman for whom the knowledge base stores some probabilistic facts. Such kind of statements are useful in particular for representing family relationships between women at risk. Finally, in order to support meaningful risk assessment via entailment one should carefully preserve generic consistency of the probabilistic ontology which can be problematic when combining findings from different studies.

P-$\mathcal{SROIQ}$ Ontology as a Theory of Breast Cancer The current amount of knowledge about breast cancer is overwhelming. For example, a meta-study conducted in 2006 by Key et al. [107] covered 98 *unique* studies focused only on the impact of a single risk factor, alcohol consumption. At the same time there are no common knowledge bases which would combine and formally represent findings produced by the multitude of studies.[6] This makes it difficult to have a global view of breast cancer risk factors and, consequently, develop tools like risk assessment calculators.

P-$\mathcal{SROIQ}$ can be used to represent a general knowledge about breast cancer in the form of a probabilistic ontology. In contrast to the risk assessment ontology, such ontology need not support risk entailments for various combinations of risk factors. Instead, its main goal is to formally and unambiguously describe the background theory of breast cancer embracing as many reliable findings as possible and serving as a common knowledge base for more specific tools, such risk assessment calculators or decision support systems.

The common breast cancer (BRC) ontology must also provide OWL vocabulary and describe non-probabilistic background knowledge of breast cancer. However, it is different from the risk assessment ontology in two key aspects. First, both its classical and probabilistic components are wider in scope. The OWL terminology is more comprehensive and provides medical vocabulary which is typically not needed in risk calculators, for example, histology-based classification of breast cancers. The probabilistic part should contain knowledge that is only implicitly represented on risk calculators such as relationships between risk factors or alternative mechanisms of how certain risk

[6] There are some lower level databases, such as ROCK—a cancer specific functional genomic database [172]. However, they do not explicitly represent case study findings and do not support such services as risk assessment.

factors increase the risk. Second, it is not entailment-oriented. Its aim is to cover the maximum number of studies so it may well be inconsistent. It does not have to be restricted to established risk factors and may represent controversial outcomes.

The set of use cases for the BRC ontology is also wider than for the BCRA ontology. In addition to maintaining a birds-eye view of breast cancer, it may be used for finding and analyzing inconsistencies in outcomes of different studies. This can be done by the conflict finding algorithm described in Section 5.2.1. It can support studying mechanisms of interactions between risk factors, for example, how alcohol consumption affects estrogen level. Finally, it may play a useful role in planning and coordination of future medical studies by helping to identify the most controversial or insufficiently studied risk factors or suspected exposures.

BCRA Ontology Design

Construction of such an ontology goes far beyond this thesis. Our contribution is to construct a small fragment of it to illustrate some useful principles of capturing domain statistics in probabilistic ontologies. We first sketch the classical part of the ontology and then proceed to interesting examples of statistical knowledge which can be represented.

Classical Part As mentioned above the classical (OWL) part provides a medical vocabulary which can be used on its own in a variety of applications or used in the representation of probabilistic knowledge. Our goal is not to create a re-usable breast cancer ontology, so we focus on providing an OWL terminology for probabilistic statements. The ontology contains the following main taxonomies:

Taxonomy of breast cancers Breast cancer is a heterogeneous disease. Some risk factors can be associated with increase in risk of developing one particular type of breast cancer and not the other. Thus it is important to classify types of breast cancer. In particular, our ontology distinguishes breast cancers by hormone receptor status. Estrogen and progesterone positive breast cancers are modeled using concepts `ERPositiveBRC` and `PRPositiveBRC` while their complements are modeled using `ERNegativeBRC` and `PRNegativeBRC` (we use shorthands ER+/- and PR+/- with obvious meaning.). Another important classification is based on histology. The ontology distinguishes between invasive and non-invasive (e.g. in situ) cancers.

Taxonomy of risk factors Dozens of risk factors are known so far. Some are established and strongly associate with increased risks, such as BRCA1(2) gene mutations, while others are controversial. The ontology should provide vocabulary for both to support current and future findings. It includes a taxonomy

of concepts rooted at `RiskFactor`. We distinguish between known risk factors (those which can be reported via a questionnaire, such as alcohol intake) and inferred risk factors which require medical examination.

Taxonomy of risks The ontology differentiates absolute and relative risks of developing breast cancer. Absolute risks are further divided into the lifetime risk and the short-term risk. Relative risks are divided into increased and reduced risks. Level of increases is a continuous variable which requires discretization (see below).

These two taxonomies induce the corresponding classification of women, i.e., classes of women w.r.t. risk factors and w.r.t. risk. For example, any risk factors `RF` gives rise to a class of women $\texttt{Woman} \sqcap \exists \texttt{hasRiskFactor.RF}$. Women having various combinations of risk factors are modeled as conjunctive concept expressions. Analogously, given a certain kind of risk `R` the expression $\texttt{Woman} \sqcap \exists \texttt{hasRisk.R}$ models those women who are in the risk group `R`, for example, have moderately increased risk of developing ER+ breast cancer. These taxonomies of women may or may not be explicitly present in the ontology. In other words, it is possible, but not essential, to generate a concept name for each interesting class of women since P-$\mathcal{SROIQ}$ (and our reasoner Pronto) allows for complex concept expressions in conditional constraints.

Probabilistic Part The probabilistic part represent statistical background knowledge about breast cancer. We currently distinguish between knowledge which explicitly associates categories of women with specific risk factors with risk and more general statistical relationships which are not necessarily risk related. We begin with the latter.

General statistical knowledge mostly includes relationships between various risk factors. For example, Ashkenazi Jew women are more likely to develop BRCA gene mutations, while early menarche, late first child (or no live births), lack of breastfeeding and alcohol consumption all increase levels of estrogen in blood. Such relationships are important because they can help to infer the presence of some risk factors given the set of known factors. They are typically easy to represent by using conditional constraints of the form $(\texttt{Woman} \sqcap \exists \texttt{hasRiskFactor.RFY} | \texttt{Woman} \sqcap \exists \texttt{hasRiskFactor.RFX})\texttt{[l,u]}$ which says that the chances of having risk factor RFY given RFX are between l and u. One possible source of complications is continuous variables, e.g. the level of estrogen, which are discussed below.

Most of statistical findings available in medical literature quantitatively describe risk increase for categories of women with specific risk factors. Typically such findings are presented by giving estimated parameters of a probability distribution where the random variable is the relative risk of a random woman in the population. Such parameters include the estimated mean value and the estimated confidence interval. Table 3.1 presents an example of the reported association between alcohol intake and

the risk increase among postmenopausal women taken from [175]. There are two main difficulties with representing this kind of data in P-$\mathcal{SROIQ}$. First, the risk increase is a continuous random variable so it needs to be discretized. Second, the available language supports only conditional constraints so direct representation of probability distributions is not possible.

Table 3.1: Example of a reported association between alcohol intake and the risk of hormone receptor-specific breast cancer (excerpt from [175])

Alcohol (g)	**ER+**	**ER-**	**PR+**	**PR-**
	RR (95% CI)	**RR (95% CI)**	**RR (95% CI)**	**RR (95% CI)**
0	1.00 (ref)	1.00 (ref)	1.00 (ref)	1.00 (ref)
≤4	1.06 (0.91 - 1.22)	1.40 (1.00 - 1.96)	1.04 (0.89 - 1.23)	1.24 (0.95 - 1.62)
≥4	1.07 (0.90 - 1.26)	1.64 (1.14 - 2.35)	1.12 (0.93 - 1.34)	1.28 (0.96 - 1.71)

Discretization of a continuous variable is relatively straightforward. We introduce a set of disjoint concept names each of which models women in the corresponding group of risk. Specifically, we define concepts `WomenAtWeakRisk`, `WomenAtModerateRisk` and `WomenAtHighRisk` which correspond to women whose relative risk of breast cancer is weakly, moderately or strongly increased respectively. Note, that OWL 2 provides datatype support to describe the exact boundaries. We have chosen ranges (1, 1.5], (1.5, 3.0] and (3.0, +inf) for weak, moderate and strong increase in risk.

The inability to represent distributions is a more severe limitation. It leaves the modeler with the only option of *approximating* the (continuous) distribution using a finite set of points. In other words, each distribution, for example, risk increase for women consuming a certain amount of alcohol, can be approximated by specifying the probability that a *randomly* taken woman with the given exposure belongs to a specific group of risk, i.e. `WomenAtWeakRisk`, `WomenAtModerateRisk` or `WomenAtHighRisk`. This is precisely the semantics of conditional constraints in P-$\mathcal{SROIQ}$.

Assuming that the random variable is real-valued, a standard way of approximating a continuous distribution is to take each interval and compute the probability that the variable takes on a value in that interval. Then the approximation of a distribution $Pr(x)$ w.r.t. a finite set of intervals U is simply a function $\hat{Pr}$ such that $\hat{Pr}(U_i) = \int_{U_i} Pr(x)\mathrm{d}x$.

Unfortunately, such approximation of results of statistical experiments is unsatisfactory because it maps every interval to a single point. The problem is that any difference, even arbitrarily small, between two or more sampling distributions will results in conflicting probabilistic statements for every interval (because the point-valued probabilities will be different) even though the results can confirm each other from a

purely statistical point of view. Consequently this approach does not support representation and reasoning about results (or distributions) reported by multiple studies, which is essential if the BCRA ontology is to be viewed as a theory of breast cancer.

The issue of representing results of statistical experiments in P-$\mathcal{SROIQ}$ seems to be of the general nature and deserves standalone discussion. We assume a population G of size N_G and a random variable X which is normally distributed across G. We also make the realistic assumption that G is large enough so that evaluating X for all members of G is not feasible. A common approach is to take one or more random samples from G, evaluate X for them and estimate the actual distribution over G based on the sampling distributions. We use μ, σ to denote the mean and the variance of the real distribution and $\overline{X^{(i)}}, S^{(i)}$ for the mean and the variance of the sample $X^{(i)}$.

Our goal is to approximate sampling distributions in P-$\mathcal{SROIQ}$ in a *statistically coherent* way. Informally it means that satisfiability of probabilistic formulas (conditional constraints) representing two or more sampling distributions must agree with their mutual statistical consistency, i.e., whether they support a common statistical hypothesis. The hypothesis, in this case, is that there exists a distribution (not necessarily a unique one) over G with parameters μ, σ such that it is supported by all sampling distributions with the required level of confidence.

A common approach for comparing two or more sampling distributions is based on statistical hypothesis tests. For example, given two normal distributions $\overline{X^{(1)}}, S^{(1)}$, $\overline{X^{(2)}}, S^{(1)}$ a frequentist statistician would take $\overline{X^{(1)}} - \overline{X^{(2)}}$, which is a normally distributed random variable, and do a *z-test* (or a Student's t-test depending on the sample sizes) to see if the difference can be taken as 0 with the required level of confidence. It amounts to calculating standard errors of the mean (SE) for both distributions and then computing the difference *in units of SE*. Finally, if the probability of observing such difference given the null hypothesis,[7] which can be found in standard tables, is low enough, for example, ≤ 0.05, the statistician would accept the hypothesis that both distributions are consistent.

Our approach is slightly different from the outlined above. It is not based on tests but on *confidence regions* for sampling distributions. The approach, which generalizes confidence intervals and dates back to Mood [150], is to estimate a region $\mathcal{R}_\gamma$ in the parameter space for (μ, σ^2) such that on average it will contain the μ, σ^2 pair of the real distribution $100(1-\gamma)\%$ times as the number of estimations goes to infinity. More formally, a $100(1-\gamma)\%$ confidence region $\mathcal{R}_\gamma$ is a *random set* for parameters (μ, σ^2) based on a group of independent normally distributed variables X (i.e., a sample) such that [7]:[8]

[7] The null hypothesis is a default position which, in this case, could be that the population mean is different from at least one of $\overline{X^{(1)}}, \overline{X^{(2)}}$.

[8] We deliberately leave out a precise definition of random set. For the purposes of this thesis it is

$$P((\mu, \sigma^2) \in \mathcal{R}_\gamma) = 1 - \gamma, \text{ for all } (\mu, \sigma^2) \tag{3.2}$$

Informally, the confidence region specifies how far sampling distributions can deviate from the population distribution while supporting it with $100(1-\gamma)\%$ confidence. Following Mood [150] we will show that for the normal distribution the region is a convex set and, therefore can be represented by boundary values of (μ, σ^2) such that *any* sampling distribution inside the boundary will be consistent with the current distribution.

Consider the sample $X_1, \ldots, X_n$ where all X_i are independent random variables with the normal distribution $(N(\mu, \sigma^2))$. Then $\overline{X} = \frac{1}{n}\sum_{i=1}^n X_i$ and $S^2 = \frac{1}{n-1}\sum_{i=1}^n (X_i - \overline{X})^2$, i.e., the sample mean and the sample variance, are random variables. It is well known that $\overline{X}$ has the normal distribution $N(\mu, \frac{\sigma^2}{n})$ (or, equivalently, $\frac{\overline{X}-\mu}{\sigma/\sqrt{n}} \sim N(0,1)$) while $(n-1)S^2/\sigma^2$ has the chi-square distribution with $n-1$ degrees of freedom [150].

The standard tables for $N(0,1)$ and χ^2_{n-1} provide numbers a, b, c such that for fixed p_1, p_2 the following equalities hold [7]:

$$P(-a < \frac{\overline{X} - \mu}{\sigma/\sqrt{n}} < a) = p_1,$$
$$P(b < (n-1)S^2/\sigma^2 < c) = p_2$$

The crucial fact is that the two random variables are independent (see [150] for a proof) which implies that:

$$p_1 p_2 =$$
$$P(-a < \frac{\overline{X} - \mu}{\sigma/\sqrt{n}} < a, b < \frac{(n-1)S^2}{\sigma^2} < c) =$$
$$P(\overline{X} - a\frac{\sigma}{\sqrt{n}} < \mu < \overline{X} + a\frac{\sigma}{\sqrt{n}}, \frac{(n-1)S^2}{c} < \sigma^2 < \frac{(n-1)S^2}{b})$$

Therefore, the $100(p_1)(p_2)\%$ confidence region for (μ, σ^2) takes the following form:

$$\mathcal{R}_{p_1,p_2}(\overline{X}, S) = \Big\{(\mu, \sigma^2) : \ \overline{X} - \alpha\frac{\sigma}{\sqrt{n}} < \mu < \overline{X} + \alpha\frac{\sigma}{\sqrt{n}}, \tag{3.3}$$
$$\frac{(n-1)S^2}{\gamma} < \sigma^2 < \frac{(n-1)S^2}{\beta}\Big\}$$

sufficient to think of a random set as of a random variable which takes on subsets of some space.

Figure 3.1 shows the joint confidence region $\mathcal{R}$ in the parameter space (μ, σ^2). Note that it is possible, although technically messy, to generalize the definition (3.3) to the case of several independent sampling distributions. The simultaneous confidence region for k samples $X^{(1)}, \ldots, X^{(k)}$ will be a region in the $2k$-dimensional parameter space which projections on each plane $(\mu^{(i)}, (\sigma^{(i)})^2)$ will look as (3.3). Then the notion of consistency of sampling distributions can be defined as follows (we limit the attention to two samples for clarity):

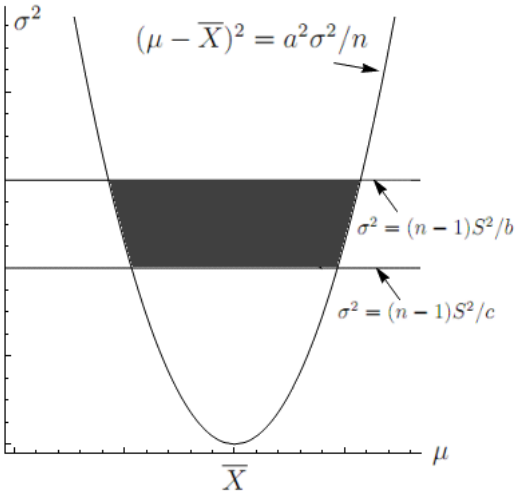

Figure 3.1: Joint confidence region for (μ, σ^2)

Definition 3.1. *Let $Pr(X^{(1)}), Pr(X^{(2)})$ be distributions on two samples $X^{(1)}, X^{(2)}$ drawn independently from a population G. They are said to be consistent with confidence $100p\%$ if there exists a point (μ, σ^2) which belongs to both $\mathcal{R}_p(\overline{X^{(1)}}, S^{(1)})$ and $\mathcal{R}_p(\overline{X^{(2)}}, S^{(2)})$.*

Now we can return to the issue of approximating a continuous sampling distribution by a discrete set of points. Assume that the domain E of a continuous real-valued random variable X is a disjoint union of a finite number of intervals $U = \{(-\infty, r_1], (r_1, r_2], \ldots, (r_{l-1}, r_l], (r_l, +\infty)\}$. Then the *approximation* of the sampling distribution $Pr(X)$ with mean and variance $(\overline{X}, S^2)$ is the function $\hat{Pr}$ which maps each interval U_i to the following real-valued set:

$$\hat{Pr}(U_i; \overline{X}, S) = \{g(\mu, \sigma^2) | (\mu, \sigma^2) \in \mathcal{R}_{p_1,p_2}(\overline{X}, S)\} \tag{3.4}$$
$$g(\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \int_{U_i} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \mathrm{d}x$$

Now we are ready to define the notion of approximate consistency of sampling distribution with respect to a set of intervals U:

Definition 3.2. *Two sampling distributions $Pr(X^{(1)}), Pr(X^{(2)})$ are approximately consistent given a finite set of intervals U if $\hat{Pr}(U_i; \overline{X^{(1)}}, S^{(1)}) \cap \hat{Pr}(U_i; \overline{X^{(2)}}, S^{(2)})$ is non-empty for all $U_i \in U$.*

As with any approximation, the utility of approximations of sampling distributions depends on what conclusions they help to draw about the distributions themselves. Given that we are interested in the matter of consistency, it is important to understand the relationships between the notions of consistency and approximate consistency of sampling distributions. The following theorem states that consistency implies approximate consistency regardless of partitioning of the real line.

Theorem 3.1. *If two sampling distributions $Pr(X^{(1)}), Pr(X^{(2)})$ are consistent, then they are approximately consistent for any choice of real-valued intervals.*

Proof. For the distribution $Pr(X^{(1)})$ a confidence region $\mathcal{R}_{p_1,p_2}(\overline{X^{(1)}}, S^{(1)})$ is connected (see Definition 3.3). The function $g(\mu, \sigma^2)$ (Definition 3.4) is continuous on it which implies that for any U_i, the set $\hat{Pr}(U_i; \overline{X^{(1)}}, S^{(1)})$ is a real-valued interval (l_1, u_1). Now consider a point $\mu_0, \sigma_0^2 \in \mathcal{R}_{p_1,p_2}(\overline{X^{(1)}}, S^{(1)}) \cap \mathcal{R}_{p_1,p_2}(\overline{X^{(2)}}, S^{(2)})$ which exists since the distributions are consistent. It follows that $l_1 < g(\mu_0, \sigma_0^2) < u_1$ (and analogously $l_2 < g(\mu_0, \sigma_0^2) < u_2$ for $\hat{Pr}(U_i; \overline{X^{(2)}}, S^{(2)})$), so $g(\mu_0, \sigma_0^2)$ is a common point for both approximations on U_i. As such the distributions are approximately consistent. □

An obvious corollary of the theorem is that inconsistency of approximations implies inconsistency of sampling distributions (the converse is false). As we demonstrate below, the inconsistency of approximations, which is based on disjointness of intervals, can be proved by logical reasoning, which means that the above result enables approximate reasoning about sampling distributions in a purely logical way. Even though the power of such reasoning is fairly limited (with consistency being the only reasoning task for the moment), its integration with OWL reasoning and the ability to use common, formally defined terminology for representation of statistical experiments is promising.

Now we present an example of approximate representation of sampling distributions in P-$\mathcal{SROIQ}$. We take two results of statistical experiments aimed at investigating associations between alcohol consumption and the increased risk of breast cancer among postmenopausal women. The findings are published in reports by Suzuki et al. [181] and Sellers et al. [175]. Unfortunately it is common for medical research papers to not explicitly present all parameters that characterize results of their statistical analyses. Typically, only the estimated mean and the confidence interval are presented while, for example, the kind of distribution is left to the reader to infer from other information

(such as the kind of regression model used). Due to that fact and because the approach above has only been developed for normal distributions, we illustrate it on an artificial example. The information given in the example is analogous to that given in medical literature, e.g. [175, 181], but is complete in the sense that all parameters and the type of sampling distributions are known.

Example 3.1. *Consider two hypothetical papers which report results of independent studies of associations between alcohol consumption among postmenopausal women and their relative risk of developing breast cancer. According to study A the mean relative risk (RR) of ER+ breast cancer for women drinking $\geq$ 4g of ethanol a day is 1.8 and has variance of 0.5. Study B has reported that the mean RR of ER+ breast cancer for the same level of drinking is 2.2 (variance 0.7). The number of cases in the studies was 230 and 150 respectively.*

We propose the following four step procedure for an approximate representation of statistical results, similar to those in the example above, in P-$\mathcal{SROIQ}$:

Preparing concepts The first step is to define the terminology used to describe the distribution, i.e., evidence and conclusion concepts. In our case evidence concepts should describe categories of women with respect to specific risk factors, e.g. alcohol intake, while conclusion concepts describe groups of women stratified by risk increase. For instance, the concept expression $\texttt{C} \equiv \texttt{Woman} \sqcap \exists \texttt{hasRiskFactor}.(\texttt{Postmenopause} \sqcap \texttt{ModerateConsumption})$ is used to model postmenopausal women with moderate level of alcohol intake.[9] On the other hand the expression:

$$D \equiv \texttt{Woman} \sqcap \exists \texttt{hasRisk}.(\texttt{ModeratelyIncreasedRisk} \sqcap \exists \texttt{riskOf}.\texttt{ERPositiveBRC})$$

models women who are at moderately increased risk of developing ER-positive breast cancer. Using these expressions the modeler can specify the probability than a random women the class `C` also belong the risk group `D` as `(D|C)[l,u]`.

Determining parameters of sampling distributions (if required) Sometimes parameters of sampling distributions can be determined from other information. For example, knowing the kind of distribution, sample mean, sample size, confidence interval and the methodology of its estimation, one can calculate the sample variance.[10] In

[9] The level of intake is a continuous variable which we also split onto categories `LimitedConsumption`, `ModerateConsumption` and `HeavyConsumption` which correspond to ≤ 4, $4 - 9.9$ and ≥ 10g of ethanol per day.

[10] The variable $T = (\overline{X} - \mu)/(S/\sqrt{n})$ has the t-dustribution with $n-1$ degrees of freedom. Confidence interval is standardly computed as $[\overline{X} - a, \overline{X} + a]$ where $a = t_{\frac{1-\alpha}{2},n-1}\frac{S}{\sqrt{n}}$ ($t_{\frac{1-\alpha}{2},n-1}$ is the $\alpha-$percentile of the Student distribution). If the confidence interval and α are known, then S can be calculated.

our case it is not needed as the distributions are normal and the parameters are known.

Choosing intervals Choice of intervals for an approximation of a continuous random variable is driven by balancing the quality of the approximation (i.e., how closely it models the continuous distribution) and the number of statements required. The latter has a direct impact on performance. For Example 3.1 we use three concepts `WomenAtWeakRisk`, `WomenAtModerateRisk` and `WomenAtHighRisk` which correspond to relative risk intervals of `(1,1.5]`, `(1.5, 3.0]` and `(3.0,`$+\infty$`)` respectively.

Computing the approximation The final (and the central) step is to compute probability intervals for the statements that approximate the continuous distribution. Each statement specifies the lower and upper probabilities that the continuous random variable X will fall into an interval U_i given that parameters of the distribution can vary within the confidence region (3.3). More formally, given the interval U_i, e.g. `(1,1.5]` for `WomenAtWeakRisk`, and the sampling distribution $(\overline{X}, S^2)$ the interval $[l_i, u_i]$ can be computed by solving the following non-linear optimization problem:

$$
\begin{aligned}
& l_i \text{ (resp. } u_i) = \ min \text{ (resp. } max) \ g(\mu, \sigma^2) \\
& s.t. \\
& (\mu, \sigma^2) \in \mathcal{R}_{p_1,p_2}(\overline{X}, S) \\
& g(\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \int\limits_{U_i} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \mathrm{d}x
\end{aligned}
\tag{3.5}
$$

In other words, $[l_i, u_i]$=$[inf\ \hat{Pr}(U_i; \overline{X}, S), sup\ \hat{Pr}(U_i; \overline{X}, S)]$.

The last preparatory step is to calculate confidence regions according to (3.3). The 95% confidence regions for distributions $(\overline{X^{(1)}}, S^{(1)}), (\overline{X^{(2)}}, S^{(2)})$ in Example 3.1 (abbreviated as $R^{(1)}_{0.95}$ and $R^{(2)}_{0.95}$) are defined by the following inequalities (calculation results rounded to three significant decimals):

$$
\begin{aligned}
R^{(1)}_{0.95} &= \left\{(\mu, \sigma^2) :\ 1.8 - \frac{2.241\sigma}{\sqrt{230}} < \mu < 1.8 + \frac{2.241\sigma}{\sqrt{230}}, 0.409 < \sigma^2 < 0.623\right\} \\
R^{(2)}_{0.95} &= \left\{(\mu, \sigma^2) :\ 2.2 - \frac{2.241\sigma}{\sqrt{150}} < \mu < 2.2 + \frac{2.241\sigma}{\sqrt{150}}, 0.548 < \sigma^2 < 0.923\right\}
\end{aligned}
$$

Now the optimization problem (3.5) can be solved numerically, e.g. by using packages such as Wolfram Mathematica, to obtain the following approximations for both sampling distributions:

$$inf\ \hat{Pr}((1,1.5];\overline{X^{(1)}},S^{(1)}) = 0.219 \qquad sup\ \hat{Pr}((1,1.5];\overline{X^{(1)}},S^{(1)}) = 0.298$$
$$inf\ \hat{Pr}((1.5,3.0];\overline{X^{(1)}},S^{(1)}) = 0.655 \qquad sup\ \hat{Pr}((1.5,3.0];\overline{X^{(1)}},S^{(1)}) = 0.878$$
$$inf\ \hat{Pr}((3.0,+\infty);\overline{X^{(1)}},S^{(1)}) = 0.239 \qquad sup\ \hat{Pr}((3.0,+\infty);\overline{X^{(1)}},S^{(1)}) = 0.586$$
$$inf\ \hat{Pr}((1,1.5];\overline{X^{(2)}},S^{(2)}) = 0.116 \qquad sup\ \hat{Pr}((1,1.5];\overline{X^{(2)}},S^{(2)}) = 0.224$$
$$inf\ \hat{Pr}((1.5,3.0];\overline{X^{(2)}},S^{(2)}) = 0.562 \qquad sup\ \hat{Pr}((1.5,3.0];\overline{X^{(2)}},S^{(2)}) = 0.769$$
$$inf\ \hat{Pr}((3.0,+\infty);\overline{X^{(2)}},S^{(2)}) = 0.189 \qquad sup\ \hat{Pr}((3.0,+\infty);\overline{X^{(2)}},S^{(2)}) = 0.568$$

So, for this example, the sampling distributions are approximately represented in P-$\mathcal{SROIQ}$ using two sets of conditional constraints:

$\{(\texttt{Woman} \sqcap \exists \texttt{hasRisk}.(\texttt{WeaklyIncreasedRisk} \sqcap \exists \texttt{riskOf}.\texttt{ERPositiveBRC})|C)[0.219, 0.298],$
$(\texttt{Woman} \sqcap \exists \texttt{hasRisk}.(\texttt{ModeratelyIncreasedRisk} \sqcap \exists \texttt{riskOf}.\texttt{ERPositiveBRC})|C)[0.655, 0.878],$
$(\texttt{Woman} \sqcap \exists \texttt{hasRisk}.(\texttt{StronglyIncreasedRisk} \sqcap \exists \texttt{riskOf}.\texttt{ERPositiveBRC})|C)[0.239, 0.586]\}$
and
$\{(\texttt{Woman} \sqcap \exists \texttt{hasRisk}.(\texttt{WeaklyIncreasedRisk} \sqcap \exists \texttt{riskOf}.\texttt{ERPositiveBRC})|C)[0.116, 0.224],$
$(\texttt{Woman} \sqcap \exists \texttt{hasRisk}.(\texttt{ModeratelyIncreasedRisk} \sqcap \exists \texttt{riskOf}.\texttt{ERPositiveBRC})|C)[0.562, 0.769],$
$(\texttt{Woman} \sqcap \exists \texttt{hasRisk}.(\texttt{StronglyIncreasedRisk} \sqcap \exists \texttt{riskOf}.\texttt{ERPositiveBRC})|C)[0.189, 0.568]\}$
where :
$\texttt{C} \equiv \texttt{Woman} \sqcap \exists \texttt{hasRiskFactor}.(\texttt{Postmenopause} \sqcap \texttt{ModerateConsumption})$

It is straightforward to verify that the approximations are consistent.

Checking consistency of sampling distributions in P-$\mathcal{SROIQ}$ may well appear cumbersome and pointless given that the same task can be done in a much simpler way and without any logical reasoning, e.g. via testing or by analyzing confidence regions. However, our aim is *not* to reduce statistical testing to logical reasoning (that aim is indeed pointless). Our aim is to represent results of statistical experiments using *common, unambiguously defined logical vocabulary* and be able to reason about them. Even though probabilistic reasoning about statistical results is quite weak at the moment, i.e., it is limited to approximate consistency checking, the potential benefits are in combining it with reasoning about the classical knowledge. For example, the BCRA ontology contains a little taxonomy of breast cancers by hormone receptor status. This enables us to combine results of the studies which are of different levels of granularity. For instance, Sellers et al. [175] report associations between alcohol intake and ER(+/-) breast cancer risk, while Suzuki et al. [181] divide it further to ER(+/-)PR(+/-) risks. In that simple case non-logical reasoning about the reported results becomes much less straightforward, while studies can also distinguish histologic types of breast cancer

(see [129]). In such complex situations reasoning about findings does involve reasoning about background knowledge, e.g. the taxonomy of breast cancers, so a combination of OWL and probabilistic reasoning is potentially beneficial.

3.2 Consistency of CADIAG-2

This section presents the methodology and the results of diagnosing the knowledge base of CADIAG-2, a large-scale medical expert system. It consists of a large collection of rules representing knowledge about various medical entities (symptoms, signs, diseases...) and relationships between them. The major portion of the rules are uncertain, i.e., they specify *to what degree* a medical entity is *confirmed* by another medical entity or a combination of them. Given the size of the system and the uncertainty, it has been challenging to validate its consistency. Recent attempts to partially formalize CADIAG-2's knowledge base into decidable Gödel logics have shown that, on that formalization, CADIAG-2 is inconsistent [33]. We verified this result with an alternative, more expressive formalization of CADIAG-2 as a set of probabilistic conditional statements and applied Pronto to determine *satisfiability* of the knowledge base and to extract conflicting sets of rules [114].[11]

To our knowledge, the probabilistic version of CADIAG-2 is the largest PSAT problem to be solved by an automated reasoner and is almost certainly the largest non-artificial one reported in the literature. This could be a bit misleading as it is comparatively easy to detect unsatisfiability by first heuristically detecting small but likely unsatisfiable fragments, and then performing a satisfiability check on each fragment. While this might suffice to validate that the knowledge base is unsatisfiable it is not sufficient, without further qualification, to detect *all* conflicting sets of rules, nor can it prove satisfiability after correcting all found conflicts (more information on computing all conflicts, or *diagnosis* of a knowledge base, can be found in Section 5.2).

As CADIAG-2 is too large (the number of rules in the binary fragment we are concerned with is over 18000) we describe an approach to split the knowledge base into comparatively large fragments that can be tested independently and prove that such methodology is complete (i.e., is guaranteed to find all conflict sets). With this methodology we are able to determine that CADIAG-2 contains many sets of conflicting rules and compute all of them for a slightly relaxed interpretation of the knowledge base.

3.2.1 Background

CADIAG-2 (Computer Assisted DIAGnosis) is a well-known rule-based expert system aimed at providing support in diagnostic decision making in the field of internal

[11] This section is largely based on our publications: [114] and [115].

medicine. Its design and construction was initiated in the early 80's at the Medical University of Vienna by K.P. Adlassnig (a detailed information on the origins and design of CADIAG-2 can be found in [4, 3, 2, 128]). It consists of two fundamental pieces: the *inference engine* and the *knowledge base*. The inference engine is based on methods of approximate reasoning in fuzzy set theory, which led some authors to present CADIAG-2 as an example of a fuzzy expert system [117, 191].

The knowledge base consists of a set of *IF-THEN* rules, known in the literature as *production* rules, which intend to represent relationships between distinct medical entities: symptoms, findings, signs and test results on the one hand and diseases and therapies on the other. In this work we consider only *binary* rules (i.e., they relate single medical entities), which are the vast majority and the only rules used by the inference engine of CADIAG-2. Consider the following example of a binary rule (taken from [2]):

```
IF suspicion of liver metastases by liver palpation
THEN pancreatic cancer
with degree of confirmation 0.3
```

The *degree of confirmation* refers, intuitively, to the degree to which the antecedent (i.e., '*suspicion of liver metastases by liver palpation*' in the example above) confirms the consequent (i.e., '*pancreatic cancer*' above). A part of our contribution is the development of a formal, probabilistic formalization of such degrees (see also [164]).

3.2.2 Probabilistic Formalization

In the following we will be working with a finite set $\mathcal{L} = S \cup D = \{P_1, ..., P_n\}$ of unary predicates in a first-order language, for some $n \in \mathbb{N}$. They represent the set of medical entities occurring in the CADIAG-2 rules, with S the set of symptoms, findings, signs and test results (to which we will commonly refer as *symptoms*) and D the set of therapies and diseases (to which we will commonly refer as *diseases*).

Definition 3.3 (CADIAG-2 Interpretation)**.** *An interpretation $\mathcal{I}$ of $\mathcal{L}$ is a pair $(\Delta^{\mathcal{I}}, V^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is a finite, non-empty set (the domain) and $V^{\mathcal{I}}$ is a map from $\mathcal{L} \times \Delta^{\mathcal{I}}$ to $[0,1]$. It is said to be **classical** (resp. **rational**) if $V^{\mathcal{I}}(P,a) \in \{0,1\}$ (resp. $V^{\mathcal{I}}(P,a) \in [0,1] \cap \mathbb{Q}$) for all $(P,a) \in \mathcal{L} \times \Delta^{\mathcal{I}}$.*

The set of binary rules in CADIAG-2, denoted as Φ_{CB}, can be classified into three different types: rules in which both antecedent and consequent are medical entities in S (*symptom-symptom*, $\Phi_{S|S}$), rules in which both antecedent and consequent are medical entities in D (*disease-disease*, $\Phi_{D|D}$) and those in which the antecedent is a medical

entity in S and the consequent an entity in D (*symptom-disease*, $\Phi_{D|S}$).[12] The degree of confirmation in a rule of the first two types either 0 or 1 and for that reason we call them *classical* while others are called *probabilistic*.

Let $\langle P, Q, \eta\rangle \in \Phi_{CB}$ be a binary rule in CADIAG-2, with $P, Q \in \mathcal{L}$ and $\eta \in [0,1] \cap \mathbb{Q}$. The value η is intended to quantify the *degree* to which P (the antecedent) *confirms* Q (the consequent) and claimed in most of the literature on CADIAG-2 (see, for example, [3] or [2]) to have been calculated from a certain database or interpretation $\mathcal{I}$ as follows:[13]

$$\frac{\sum_{a\in\Delta^{\mathcal{I}}} min\{V^{\mathcal{I}}(P,a), V^{\mathcal{I}}(Q,a)\}}{\sum_{a\in\Delta^{\mathcal{I}}} V^{\mathcal{I}}(P,a)} = \eta \tag{3.6}$$

This expression generalizes the frequentist's notion of *conditional probability* that one gets when restricting the model to classical interpretations. Under such a restriction η becomes a probability (in the sense of frequency) and its meaning is intuitive and formally well understood. However, whether the interpretation is assumed in terms of *any* valuations or in terms of only *classical* valuations (i.e., a probabilistic interpretation) will be indifferent to the purpose of finding conflicts in CADIAG-2.

The expression $Q/_{\mathcal{I}}P = \eta$ will be used to abbreviate (3.6). Sometimes, in order to generalize results, we will be considering an interval, say $\Omega \subseteq [0,1]$, instead of a single value. In that case the expression $Q/_{\mathcal{I}}P \in \Omega$ will abbreviate the corresponding modification of (3.6). Such modification is motivated by the possibility of alternative, suitable interpretations of the rules in Φ_{CB} that one could consider interesting from theoretical or practical perspectives. In particular, η in equation (3.6) can be replaced by the interval $[\eta, 1]$ (i.e., consider η a lower bound for the degrees of confirmation instead of a precise one) or by an interval of the form $[\eta - \epsilon, \eta + \epsilon]$, for a small ϵ (i.e., a slightly relaxed interpretation of Φ_{CB}).

We will denote the collection of real intervals in $[0,1]$ by $\mathfrak{I}$. We will normally refer to intervals of the form $[\eta, \eta] \in \mathfrak{I}$ by η itself. For the next definition and the lemma let $\mathcal{I}$ be an interpretation of $\mathcal{L}$ and $\Phi \subseteq \mathcal{R}_{\mathcal{L}}$, for $\mathcal{R}_{\mathcal{L}} = \{\langle P, Q, \Omega\rangle \mid P, Q \in \mathcal{L}, \Omega \in \mathfrak{I}\}$.

Definition 3.4 (CADIAG-2 Model)**.** *$\mathcal{I}$ is a model of Φ, denoted as $\mathcal{I} \models \Phi$, if $Q/_{\mathcal{I}}P \in \Omega$ for all $\langle P, Q, \Omega\rangle \in \Phi$.*

Lemma 3.1 (Classical and Rational Satisfiability, see [114])**.** *The following statements*

[12] CADIAG-2 also contains rules with a medical entity in D as the antecedent and a medical entity in S as the consequent. However, such rules are not used by the CADIAG-2 inference mechanism and ignored in our formalization.

[13] There are some references in which the interpretation suggested for η in $\langle P, Q, \eta\rangle$ is different. For example in [4] it is claimed that η can be interpreted as a frequency and thus $\langle P, Q, \eta\rangle$ as a probabilistic conditional statement.

are pairwise equivalent: i) Φ has a model (or Φ is consistent), ii) Φ has a rational model, and iii) Φ has a classical model.[14]

Lastly, we define what is meant by a minimal conflict in the CADIAG-2 rule base. Obviously, there is a direct connection between conflicts in CADIAG-2's KB and the general notion of minimal conflicts in P-$\mathcal{SROIQ}$'s PTBoxes, which are defined in Section 5.2.1.

Definition 3.5. *A set of rules $\Phi \subseteq \mathcal{R}_{\mathcal{L}}$ is a **minimal unsatisfiable set** (or minimal inconsistent set) if it is not satisfiable and, for all $\Phi^* \subset \Phi$, Φ^* is satisfiable.*

Now we are ready proceed to the translation of CADIAG-2 KB into P-$\mathcal{SROIQ}$.

Definition 3.6 (Translation into P-$\mathcal{SROIQ}$)**.** *The CADIAG-2 KB, Φ_{CB}, is translated into a PTBox $PT_{CB} = (\mathcal{T}, \mathcal{P})$ as follows:*

$$\begin{aligned}\mathcal{T} =&\{P \sqsubseteq Q | \langle P, Q, \Omega\rangle \in \Phi_{D|D} \cup \Phi_{S|S}, \Omega = \{1\}\} \cup \\ &\{P \sqcap Q \sqsubseteq \bot | \langle P, Q, \Omega\rangle \in \Phi_{D|D} \cup \Phi_{S|S}, \Omega = \{0\}\} \\ \mathcal{P} =&\{(Q|P)[l, u] | \langle P, Q, \Omega\rangle \in \Phi_{D|S}, \Omega = [l, u]\}\end{aligned}$$

Informally, we use a one-to-one mapping between concept names in $\mathcal{SROIQ}$ and unary predicates in the CADIAG-2 vocabulary $\mathcal{L}$. Each classical rule in Φ_{CB} is translated into a TBox axiom (concept inclusion or disjointness, depending on whether the degree of confirmation is 1 or 0 respectively) and each probabilistic rule is translated into a conditional constraint. The following theorem, which is a direct consequence of Proposition 3 in [114], establishes faithfulness of this translation:

Theorem 3.2 (The Translation Is Faithful)**.** *Φ_{CB} has a model if and only if PT_{CB} is a coherent PTBox.*

3.2.3 Diagnosis and Results

The theorem 3.2 enables us to reduce the problem of finding conflicting sets of rules in CADIAG-2, to the problem of finding minimal sets of conflicts in a PTBox. This reasoning problem is known as the *Diagnosis* problem and is formally defined in Section 5.2.1. The only subtle point is that consistency of CADIAG-2 KB is equivalent to *coherency*, not satisfiability, of the corresponding PTBox. This is due to the semantics of P-$\mathcal{SROIQ}$ which allows for vacuous satisfiability of conditional constraints by interpretations that assign zero probability to the evidence (see Section 2.3.4 for some discussion of this issue). Therefore, for practical purposes, we will check approximate

[14] Proof of this lemma as well as Proposition 3 in [114] and Lemma 3.2 are attributed to David Picado-Muiño.

coherence of PT_{CB} by solving PSAT and Diagnosis for the augmented PTBox $PT_{CB}^{0.001}$ (see Definition 2.21 and Theorem 2.1), in which the probability of every symptom is asserted to be equal to or greater than 0.001.

The number 0.001 is in this case domain-specific. No degree of confirmation in CADIAG-2 can be equal to or less than 0.001 as it is used to indicate that the actual degree in a rule is unknown. Consequently, the case when PT_{CB} is satisfiable but not 0.001-coherent, i.e., satisfiable *only* when probability of some symptom is less than 0.001, can be regarded as a modeling error that is essentially equivalent to inconsistency.[15] Therefore, our approach to the inconsistency analysis of CADIAG-2 is to solve the Diagnosis problem for $PT_{CB}^{0.001}$.

Decomposition

To our knowledge, none of the existing probabilistic solvers, including propositional ones, can solve PSAT for the whole of Φ_{CB} within reasonable amount of time (see Table 1 for a precise account of the size of Φ_{CB}). However, it has a certain structure that allows its *decomposition* into fragments that can be examined independently. A crucial property of our probabilistic formalization of CADIAG-2 is that Φ_{CB} is satisfiable if and only if all of the fragments are individually satisfiable, as we show below.

Table 3.2: Characteristics of CADIAG-2's knowledge base

Number of distinct symptoms	1761
Number of distinct diseases	341
Number of symptom-symptom rules (size of $\Phi_{S\|S}$)	720
Number of disease-disease rules (size of $\Phi_{D\|D}$)	218
Number of symptom-disease rules (size of $\Phi_{D\|S}$)	17573

Φ_{CB} can be represented as a directed graph where the nodes are the medical entities in $\mathcal{L}$ and the edges are specified by the rules in Φ_{CB} (i.e., a rule of the form $\langle P, Q, \eta \rangle$ in Φ_{CB} would correspond to an edge directed from P to Q). Given $P \in \mathcal{L}$, we write $\Phi_P \subseteq \Phi_{CB}$ to denote the set of rules that yield a directed edge in a path from P to any other medical entity in $\mathcal{L}$ or a directed edge in a path from any medical entity in $\mathcal{L}$ to P. Then, due to the simple structure of CADIAG-2, the following statements hold (see proofs of Proposition 4 and Corollary 2 in [114]):

[15] Leaping ahead, none of the discovered conflicts can be eliminated by refining the approximation so 0.001 proved to be a good choice in the case of CADIAG-2.

Lemma 3.2 (Decomposition's Properties)**.** *Let $P_1, P_2 \in S$ be two medical entities such that there is no path from P_1 to P_2 in Φ_{CB} or vice versa and that there is no medical entity $P \in \mathcal{L}$ from which there exists a path both to P_1 and P_2. Then:*

i) If Φ_{P_1} and Φ_{P_2} are satisfiable then $\Phi_{P_1} \cup \Phi_{P_2}$ is satisfiable.

ii) If Φ is a minimal unsatisfiable set of rules in $\Phi_{P_1} \cup \Phi_{P_2}$ then either $\Phi \subseteq \Phi_{P_1}$ or $\Phi \subseteq \Phi_{P_2}$.

Based on Theorem 3.2 we split Φ_{CB} into a set of fragments of the form Φ_P, where $P \in S$ is a symptom such that there is no rule in Φ_{CB} of the form $\langle Q, P, \eta \rangle$. For simplicity we include the entire $\Phi_{D|D}$ in each fragment since it is decomposable to a much less extent than $\Phi_{S|S}$. The largest fragments have around 200 probabilistic formulas that normally relate two or three connected symptoms to diseases.

Note that in general i) in Theorem 3.2 does not imply ii), in other words, a satisfiability preserving decomposition of an arbitrary KB does not ensure that the union of conflict sets for both fragments is equivalent to the set of conflicts of the whole KB. For example, when both fragments are unsatisfiable (and so is their union) there could be the third conflict which contains rules from both fragments, so it can only be found by analyzing the whole KB. The second claim in Theorem 3.2 means that this is not possible if CADIAG-2's KB is decomposed according to the criteria above. Consequently, the approach of computing conflicts on each fragment is *complete*. These properties hold for the formalization of CADIAG-2, including the one where intervals are used in place of point-valued probabilities, due to the following features of the rule base:

P1 All formulas contain only atomic medical entities (i.e., entities in $\mathcal{L}$).
P2 All probabilistic formulas in $\Phi_{D|S}$ condition only on symptoms (uncertain rules are unidirectional).
P3 The graph of $\Phi_{S|S}$ contains numerous disconnected components.

Next we present the results of finding conflicts in the decomposed CADIAG-2 KB.

Results

We present here results concerning the consistency check of Φ_{CB} when considering a slightly relaxed interpretation of Φ_{CB} by replacing each rule of type *symptom-disease* of the form $\langle P, Q, \eta \rangle \in \Phi_{CB}$, for some $P, Q \in \mathcal{L}$ and $\eta \in (0,1) \cap \mathbb{Q}$, by $\langle P, Q, \Omega_\eta \rangle$, with

$$\Omega_\eta = [\eta - 0.01, \eta + 0.01] = [\eta^-, \eta^+].^{16}$$

We have opted for checking consistency of this slightly relaxed interpretation of

[16] The degrees of confirmation of the rules in Φ_{CB} are all of the form $\frac{k}{100}$, for some $k \in \{0, 1, ..., 100\} \subset \mathbb{Z}$. Thus Ω_η is well defined.

the rules in Φ_{CB} against a precise interpretation (i.e., the standard interpretation with precise values) because of time constraints. The implementation of our algorithms for the relaxed interpretation of Φ_{CB} completes the task of finding all minimal unsatisfiable subsets in a reasonable amount of time (around one hour). It is a well-known fact in model-diagnosis theory that computing *all* minimal unsatisfiable subsets of a certain knowledge base requires a number of satisfiability tests (in our case, PSAT tests) that is (in the worst case) exponential in the number of unsatisfiable subsets. Our relaxed interpretation of Φ_{CB} already contains a high number of unsatisfiable sets (as we will just see) and a precise interpretation (being stronger) induces still more. Furthermore, some of the unsatisfiable sets that are present in the precise interpretation and not in our relaxed one are relatively large (some contain 7 rules) and do not overlap with other unsatisfiable sets. Such facts bring the algorithm's running time closer to its worst case.

An example of a type of minimal unsatisfiable set detected under a precise interpretation of the rules but not under our relaxed version is the one that follows:

$$\langle P_1, Q_1, \eta_1\rangle,\ \ \langle P_1, Q_2, \eta_2\rangle\ \ \langle P_2, Q_1, \eta_3\rangle\ \ \langle P_2, Q_3, \eta_4\rangle,$$

$$\langle Q_1, Q_3, 1\rangle\ \ \langle Q_2, Q_3, 1\rangle\ \ \langle P_1, P_2, 1\rangle,$$

for $P_1, P_2 \in S$, $Q_1, Q_2, Q_3 \in D$, $\eta_1, \eta_2, \eta_3, \eta_4 \in [0,1]$, with $\eta_3 = \eta_4$ and $\eta_1 < \eta_2$. Notice that the rules $\langle P_2, Q_1, \eta_3\rangle$ and $\langle P_2, Q_3, \eta_4\rangle$ along with $\langle Q_1, Q_3, 1\rangle$ intuitively claim that the set of patients with symptom P_2 and disease Q_2 coincides with the set of patients with symptom P_2 and disease Q_3 when assuming $\eta_3 = \eta_4$. Under such an assumption the rules $\langle P_1, Q_1, \eta_1\rangle$ and $\langle P_1, Q_2, \eta_2\rangle$ along with the remaining classical rules generate an inconsistency whenever $\eta_1 < \eta_2$. Notice also that, for example, for $\eta_3 < \eta_4$ the set would not be unsatisfiable and thus our relaxed interval interpretation would yield this set consistent (assuming $\eta_3, \eta_4 < 1$).

For the sake of simplicity we adopt the same notation for the rules of type *symptom-disease* of the form $\langle P, Q, \eta\rangle$, with $\eta \in \{0,1\}$. We will write $\langle P, Q, \Omega_\eta\rangle$, with $\Omega_\eta = [\eta, \eta] = [\eta^-, \eta^+]$.

Next we present the different types of minimal unsatisfiable sets encountered in Φ_{CB} under this relaxed interpretation of the rules. The statistics regarding the number of occurrences of each type and the corresponding number of rules is shown is Table 3.2.3.

Type 1. Our first type of minimal unsatisfiable set in Φ_{CB} is given by a collection of rules of the form

$$\langle P, Q_1, \Omega_\eta\rangle,\ \ \langle P, Q_2, \Omega_\zeta\rangle,\ \ \langle Q_1, Q_2, 1\rangle,$$

for $P \in S$, $Q_1, Q_2 \in D$, $\eta, \zeta \in [0,1]$ and $\zeta^+ < \eta^-$.

By $\zeta^+ < \eta^-$ we are intuitively assuming that the number of patients that have both

Table 3.3: The size and the number of minimal unsatisfiable sets of various types in Φ_{CB} under the relaxed interpretation.

Type of minimal conflict	Number of conflicts	Number of rules involved
Type 1	420	3
Type 2	5	3
Type 3	1	3
Type 4	269	6

symptom P and disease Q_1 is greater than the number of patients with both symptom P and disease Q_2, which contradicts $\langle Q_1, Q_2, 1\rangle$ (i.e., the assumption that all patients that have disease Q_1 have also disease Q_2).

Type 2. Our second type of minimal unsatisfiable set in Φ_{CB} is given by a set of rules of the form

$$\langle P, Q_1, \Omega_\eta\rangle,\ \ \langle P, Q_2, \Omega_\zeta\rangle,\ \ \langle Q_1, Q_2, 0\rangle,$$

for $P \in S$, $Q_1, Q_2 \in D$, $\eta, \zeta \in [0, 1]$ and $\eta^- + \zeta^- > 1$.

Notice that the rule $\langle Q_1, Q_2, 0\rangle$ assumes *disjointness* between Q_1 and Q_2 (intuitively, there cannot be a patient with both disease Q_1 and Q_2), which rules out the possibility of consistency whenever $\eta^- + \zeta^- > 1$.

Type 3. The third type of minimal conflict set in Φ_{CB} is given by a set of the form

$$\langle P_1, Q, \Omega_\eta\rangle,\ \ \langle P_2, Q, \Omega_1\rangle,\ \ \langle P_1, P_2, 1\rangle,$$

for $P_1, P_2 \in S$, $Q \in D$, $\eta \in [0, 1]$ and $\eta^+ < 1$.

Intuitively, the rule $\langle P_1, P_2, 1\rangle$ says that all patients with symptom P_1 also have symptom P_2. The rule $\langle P_2, Q, \Omega_1\rangle$ intuitively says that all patients with symptom P_2 have disease Q. These two facts together imply that patients with symptom P_1 should all have disease Q (i.e., $\eta^+ = 1$).

Type 4 The fourth and last type of minimal unsatisfiable set is given by a collection of rules of the form

$$\langle P, Q_1, \Omega_\eta\rangle,\ \ \langle P, Q_2, \Omega_\zeta\rangle,\ \ \langle P, Q_3, \Omega_\lambda\rangle\ \ \langle Q_1, Q_3, 1\rangle,\ \ \langle Q_2, Q_3, 1\rangle,\ \ \langle Q_1, Q_2, 0\rangle,$$

with $P \in S$, $Q_1, Q_2, Q_3 \in D$, $\eta, \zeta, \lambda \in [0, 1]$, $\lambda^+ < \eta^- + \zeta^- \leq 1$ and $\zeta^-, \eta^- \leq \lambda^+$ (to guarantee minimality).

Intuitively, assuming $\langle P, Q_1, \Omega_\eta\rangle$, $\langle P, Q_2, \Omega_\zeta\rangle$ and $\langle Q_1, Q_2, 0\rangle$, the proportion of patients that, having symptom P, have either disease Q_1 or Q_2 is at least $\eta^- + \zeta^-$. On the other hand, assuming $\langle Q_1, Q_3, 1\rangle$ and $\langle Q_2, Q_3, 1\rangle$, we have that all patients with disease either Q_1 or Q_2 have also disease Q_3. Thus, under such assumptions, satisfiability

requires that $\lambda^+ \geq \eta^- + \zeta^-$.

Summary

Our work on CADIAG-2 presents several contributions: First, we developed a probabilistic formalization of CADIAG-2's KB as a PTBox in P-$\mathcal{SROIQ}$ which, in contrast to alternative fuzzy formalizations [33], is equisatisfiable with the original set of uncertain rules. Given completeness of our Diagnosis algorithm (Section 5.2.1) it ensures finding all minimal unsatisfiable set of rules, i.e., the complete and precise description of CADIAG-2's inconsistency.[17]

Second, we provided a decomposition scheme for CADIAG-2's KB. It is specific to CADIAG-2 but illustrates a possible approach to decomposing a large probabilistic knowledge base into fragments of feasible size. We proved two important properties of the decomposition: that it preserves satisfiability and completeness of the set of minimal conflicts. A more general approach to decomposition, which could, for example, be based on the notion of probabilistic independence [5, 154, 184], is left for future research.

Lastly, we managed to carry out a complete diagnosis of the relaxed version of CADIAG-2's KB, i.e., compute all minimal inconsistent sets of rules. A thorough analysis of these types of inconsistencies in connection with the whole knowledge base and with possible repair strategies and in relation to other sets of inconsistencies obtained under alternative interpretations of CADIAG-2 is an ongoing work [115].

We hope that CADIAG-2, or CADIAG-2 like problems, will be taken up by the PSAT solving community. Due to its structure CADIAG-2 is interestingly different from artificially generated problems while its size sets a new base line for scalable PSAT and other reasoning procedures. We report the results of evaluating our Diagnosis, PTCon and TLexEnt algorithms on fragments of CADIAG-2 in Section 7.2.

3.3 Reasoning about Ontology Alignments

This section describes another application of P-$\mathcal{SROIQ}$ and Pronto: reasoning about ontology alignments produced either manually or by certain ontology alignment tools. We briefly outline previously proposed approach to validate alignments using a probabilistic formalism and discuss different probabilistic formalizations (including the one based on Jaccard similarity). Finally, we argue that probabilistic validation can be more appropriate than previously described validation via classical DL reasoning.

[17] Here again by "complete" we mean complete for the binary fragment of CADIAG-2. Equisatisfiability is not preserved if compound rules are considered and it is unclear if it *can* be preserved under a probabilistic formalization.

3.3.1 Ontology Alignments

Ontology alignment is an important task in domains where multiple ontologies have been evolving independently. In such domains some form of ontology interoperability is typically required since otherwise ontology engineers could be left with the only unpleasant option of duplicating knowledge in several ontologies, which could easily lead to inconsistencies and propagation of errors.

Ontology alignment is an alternative to, or a step towards, ontology integration, i.e., transformation of several independent ontologies into a single, unified ontology. Alignment is beneficial in situations when it is desirable to keep the ontologies separate and able to evolve independently. Instead of merging the ontologies are *aligned* by establishing correspondences between their structural components: concepts, roles, and individuals. Such correspondences are often called *mappings*. We adopt the following widely established definition of mappings as semantic relations between components ([54, 145]):

Definition 3.7 (Ontology Mapping)**.** *Given ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$, let Q be a function that defines sets of matchable elements $Q(\mathcal{O}_1)$ and $Q(\mathcal{O}_2)$. A correspondence between $\mathcal{O}_1$ and $\mathcal{O}_2$ is a 4-tuple (e, e', r, n) such that $e \in Q(\mathcal{O}_1)$ and $e' \in Q(\mathcal{O}_2)$, r is a semantic relation between e and e', and n is a confidence value from a suitable structure $(D, \leq)$. A mapping between $\mathcal{O}_1$ and $\mathcal{O}_2$ is a set of correspondences between $\mathcal{O}_1$ and $\mathcal{O}_2$.*

In this thesis we focus on a special case of mappings among all those that can be captured by Definition 3.7. We only consider alignments of concept hierarchies since they typically occupy the central place in OWL ontologies, so the function Q returns the set of all atomic concepts in an ontology. Alternatively, Q may return the (possibly infinite) set of concept expressions over the ontology signature, which is useful if a particular tool can map complex concepts. In addition, similarly to [145], we restrict r to be one of $\{\equiv, \sqsubseteq\}$. Finally, we take the confidence structure to be the unit interval $D = [0, 1]$ with the natural total order.

It has been argued that ontology alignments can be analyzed, in particular, validated by logical reasoning, in the context of the integrated ontology containing both $\mathcal{O}_1$ and $\mathcal{O}_2$, as well as additional axioms representing mappings [145]. More formally:

Definition 3.8 (Integrated Ontology)**.** *Given ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$ of finite size, the integrated ontology $\mathcal{O}_1 \cup_{\mathcal{M}^t} \mathcal{O}_2$ of $\mathcal{O}_1$ and $\mathcal{O}_2$ connected by a mapping $\mathcal{M}$ between $\mathcal{O}_1$ and $\mathcal{O}_2$ is defined as $\mathcal{O}_1 \cup \mathcal{O}_2 \cup \{t(x) | x \in \mathcal{M}\}$, where t is a translation function that maps correspondences to axioms.*

Definition 3.8 admits different translation functions. Melicke and Stuckenschmidt consider one translation function which turns mappings into concept subsumption and equivalence axioms by simply dropping the confidence value [145].

$$t_n(C_1, C_2, r, n) = \begin{cases} C_1 \sqsubseteq C_2, & \text{for } r = \sqsubseteq \\ C_1 \equiv C_2, & \text{for } r = \equiv \end{cases}$$

In the next section we will consider another translation which preserves confidence values by representing them as probabilities. The implications of using different translations are briefly discussed in Section 3.3.3.

3.3.2 Probabilistic Formalization

It is standard for ontology aligning tools to output degrees of confidence in generated mappings. However, to the best of our knowledge, there is no widely accepted standard for the semantics of those confidence values. Informally, they reflect the confidence that a particular tool has that the relation r holds between components e and e' belonging to the ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$ respectively. Most of the tools interpret their confidence as some sort of *similarity* measure between the components without precisely defining its meaning.

However, specifying the semantics of confidence values is essential for analyzing alignments by means of logical reasoning. One example of such analysis is alignment validation and refinement whose goal is to make sure that the alignment does not contradict the logical structure of both ontologies [145, 146]. Such contradictions, which often appear in the form of unsatisfiable concepts, can cause problems for terminological reasoning with the integrated ontology as well as difficulties during data transformation and query processing (see [145, 146] for a more detailed discussion). This kind of validation is now used for assessing alignment quality during the annual Ontology Alignment Evaluation Initiative (OAEI) contest.[18]

Lack of a rigorous semantics of confidence values was not a problem for the incoherence analysis proposed by Melicke and Stuckenschmidt because they convert each mapping into a classical subsumption or equivalence axiom (possibly ignoring mappings for which confidence values are below a certain threshold). However, if confidence values are to taken into account during validation, then their interpretation needs to be specified. There are several reasons why accounting for confidence values may be beneficial for alignment analysis, namely:

- For testing and debugging mapping tools.
- For comparing mappings generated by different tools.
- For completing partial mappings via probabilistic entailment.

[18] `http://oaei.ontologymatching.org/`

In this work we focus on a particular interpretation of confidence based on probability distributions. As has been observed in [48] probability distributions over interpretations of ontology elements, such as concepts, can be used to define many practical similarity metrics thus helping to avoid committal to a particular measure. Although not all ontology alignment tools use probabilistically definable similarity measures, those which do are not uncommon. In particular, probabilistic interpretations of similarity are employed by tools which learn mappings from data or other statistical techniques, for example, GLUE [47] or OMEN [149]. Probabilistic reasoning has been suggested as a method for validating and discovering ontology alignments several times in the literature, including in [25, 29, 155].

We are mostly concerned with *validation* as a special case of logical reasoning about ontology alignments. Similarly to [29] we make an assumption that confidence n in a correspondence (C_1, C_2, r, n) is the *probability*, or the lower bound on the probability, that the relation r holds between concepts C_1 and C_2. More formally, we restrict our attention to those translation functions t (see Definition 3.8) which translate correspondences into sets of conditional constraints in P-$\mathcal{SROIQ}$. More formally:

Definition 3.9 (Probabilistic Integrated Ontology)**.** *Let $\mathcal{T}_1, \mathcal{T}_2$ be $\mathcal{SROIQ}$ TBoxes being aligned and $\mathcal{M}$ be a set of correspondences between concepts in $\mathcal{T}_1$ and $\mathcal{T}_2$. Let t_{prob} be a translation function that maps each correspondence (C_1, C_2, r, n) into a set of (conditional) constraints. Then the probabilistic integrated ontology PO is the following PTBox in P-$\mathcal{SROIQ}$: $(\mathcal{T}_1 \cup \mathcal{T}_2, \bigcup_{m \in \mathcal{M}} t_{prob}(m))$.*

However, in contrast to [29] and following the idea of Doan et al. [48] we do not commit to a particular formalization of probabilistic mappings using the language of P-$\mathcal{SROIQ}$. Instead we formulate the following simple desiderata for possible formalizations which enables probabilistic validation of ontology alignments by means of checking whether the integrated PTBox is probabilistically incoherent (i.e., $PO \models_{tight} (C|\top)[0, 0]$ for some concept C, see Section 2.3.4).

- If $\mathcal{T}_1 \cup \mathcal{T}_2 \models C_1 \sqcap C_2 \sqsubseteq \bot$ then $(\mathcal{T}_1 \cup \mathcal{T}_2, \{t_{prob}(C_1, C_2, r, n)\})$ should be incoherent for $n > 0$ for any $r \in \{\sqsubseteq, \equiv\}$.
- If $\mathcal{T}_1 \cup \mathcal{T}_2 \models C_1 \sqsubseteq C_2$ and $r = \sqsubseteq$ then $(\mathcal{T}_1 \cup \mathcal{T}_2, \{t_{prob}(C_1, C_2, r, n)\})$ should be incoherent for $n < 1$.
- If $\mathcal{T}_1 \cup \mathcal{T}_2 \models C_1 \equiv C_2$ then $(\mathcal{T}_1 \cup \mathcal{T}_2, \{t_{prob}(C_1, C_2, r, n)\})$ should be incoherent for $n < 1$ for any $r \in \{\sqsubseteq, \equiv\}$.

Intuitively, it is desirable that the probability of subsumption and equivalence would be zero for disjoint concepts and strictly one for equivalent concepts. Now we will

demonstrate that the syntax of P-$\mathcal{SROIQ}$ is powerful enough for expressing some interesting candidates for a probabilistic definition of similarity. We consider two variants for each translation function, one that interprets similarity as an exact probability and the other that interprets it as the lower bound (we write t^l to denote the latter whenever t denotes the exact version). The first is the translation function adopted from Castano et al. [29], whose exact version looks as follows:

$$t_{castano}(C_1, C_2, r, n) = \begin{cases} \{(C_2|C_1)[n,n]\}, & \text{for } r = \sqsubseteq \\ \{(C_1|C_2)[n,n], (C_2|C_1)[n,n]\}, & \text{for } r = \equiv \end{cases}$$

The lower bound version, $t^l_{castano}$, is the same except that the intervals are $[n, 1]$.

The advantage of this translation is that it uses only atomic concepts and is simple to use and understand. It is also easy to check that the exact version meets the above desiderata while the lower bound version meets only the first requirement (i.e., it is weaker). However, the main shortcoming is that the translation requires two conditional constraints for a single equivalence mapping. This means that equivalence mappings are *not* translated into expressions to which probabilities can be naturally attached in P-$\mathcal{SROIQ}$ (i.e., one will need to meta-logically combine probabilities of $(C_1|C_2)[n,n]$ and $(C_2|C_1)[n,n]$ to get a probability of $C_1 \equiv C_2$). In particular, this will complicate the entailment of the probability that the two concepts are equivalent, if that ever becomes necessary.

Another possibility is to consider concept subsumptions $C_1 \sqsubseteq C_2$ as axioms of the form $\top \sqsubseteq \neg C_1 \sqcup C_2$ and equivalence as $\top \sqsubseteq (\neg C_1 \sqcup C_2) \sqcap (C_1 \sqcup \neg C_2)$. The translation function, which we call t_u, will then translate both types of correspondences into single *unconditional* constraints:

$$t_u(C_1, C_2, r, n) = \begin{cases} \{(\neg C_1 \sqcup C_2|\top)[n,1]\}, & \text{for } r = \sqsubseteq \\ \{((\neg C_1 \sqcup C_2) \sqcap (C_1 \sqcup \neg C_2)|\top)[n,1]\}, & \text{for } r = \equiv \end{cases}$$

However, this translation suffers from the same issues as the formalization of statistical relations, e.g. "90% birds fly", as uncertain implications, e.g. $(\neg Bird \sqcup Fly)[0.9, 1]$. Even the exact version does not meet the desiderata, for example, because the concept expression $\neg C_1 \sqcup C_2$ can *coherently* have a non-zero probability even when C_1 and C_2 are disjoint. All that is necessary is that either $Pr(C_1) < 1$ or $P(C_2) > 0$.

Finally, we demonstrate the P-$\mathcal{SROIQ}$ is expressive enough to employ the Jaccard coefficient—one of the most commonly used measures of similarity [185]. It measures similarity between two sets as the size of their intersection divided by the size of their

union: $J(A,B) = \frac{|A \cap B|}{|A \cup B|}$. In P-$\mathcal{SROIQ}$ the Jaccard coefficient can be expressed by the conditional probability $P(A \sqcap B | A \sqcup B)$ which is simply an abbreviation of $\frac{P(A \sqcap B)}{P(A \sqcup B)}$ (see Section 2.3.1). This way we obtain the following translation function, which is used in our validation experiments (see Section 7.1):

$$t_{jaccard}(C_1, C_2, r, n) = \begin{cases} \{(C_2|C_1)[n,1]\}, & \text{for } r = \sqsubseteq \\ \{(C_1 \sqcap C_2 | C_1 \sqcup C_2)[n,1]\}, & \text{for } r = \equiv \end{cases}$$

Again, it is straightforward to verify that it satisfies the desiderata. It is also free of the aforementioned shortcoming of $t_{castano}$. A small disadvantage is the use of concept expressions which may have negative impact on reasoning performance (we provide more details on this issue in Section 7.1).

3.3.3 Probabilistic vs. Classical Validation

Although the idea of reasoning about confidence values during an analysis of mappings seems intuitively appealing, its exact benefits may not be immediately obvious. In particular, it may not be clear why probabilistic validation of mappings proposed by Castano et al. [29] could be superior to validation by means of classical DL reasoning proposed by Meilicke and Stuckenschmidt [145, 146].

Our argument in support of probabilistic validation or, more generally, reasoning about confidence values, is that it can produce more *accurate* results without missing any errors discoverable by the classical incoherence analysis. To be more precise, we claim that first, probabilistic incoherence of all concepts can be confirmed by the approach of Meilicke and Stuckenschmidt, and second, translation of uncertain mappings into classical DL axioms may produce *spurious* results. The second point means that ignoring confidence values, or rather treating them as fully certain axioms, could lead to unsatisfiability of some concepts which are not probabilistically unsatisfiable.

The first claim is made precise by the following theorem which states that probabilistic incoherence implies classical incoherence. Note, that in the theorem we use the lower bound versions of the translation functions presented above.

Theorem 3.3. *Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be two $\mathcal{SROIQ}$ TBoxes, $\mathcal{M}$ be a set of mappings, and $PT = (\mathcal{T}_1 \cup \mathcal{T}_2, \bigcup_{m \in \mathcal{M}} t(m))$ be a satisfiable probabilistic integrated ontology, where $t \in \{t^l_{castano}, t^l_u, t^l_{jaccard}\}$, and $T = \mathcal{T}_1 \cup \mathcal{T}_2 \cup \{t_n(m) | m \in \mathcal{M}\}$ be a satisfiable integrated ontology. Then $PT \models (X|\top)[0,0]$ for some matchable satisfiable concept X implies that $T \models X \sqsubseteq \bot$ where t_n is a natural translation of mappings into axioms.*

Proof. The proof can be found in Appendix A. □

The theorem states that if a concept becomes probabilistically unsatisfiable as a result of the alignment then it is necessarily unsatisfiable w.r.t. classical DL semantics. However, the converse is not the case as the following simple example demonstrates:

Example 3.2. *Consider* $\mathcal{T}_1 = \mathcal{T}_2 = \{A \sqsubseteq \top, B \sqsubseteq \top, C \sqsubseteq \neg B\}$, $\mathcal{M} = \{(A, B, 0.5), (A, C, 0.5)\}$. *Then classical integrated ontology* $\{A \sqsubseteq \top, B \sqsubseteq \top, C \sqsubseteq \neg B, A \sqsubseteq B, A \sqsubseteq C\}$ *entails* $A \sqsubseteq \bot$ *while PTBox* $\{\mathcal{T}_1 \cup \mathcal{T}_2, \{(B|A)[0.5, 1], (C|A)[0.5, 1]\}\}$ *does not entail* $(A|\top)[0.0]$. *Put another way, A is classically unsatisfiable according to the natural translation of mappings but is not probabilistically unsatisfiable according to either* $t^l_{castano}$, $t^l_{jaccard}$, *or* t^l_u.

Uncertain mappings in this example simply indicate that the tool has 50% confidence that A is a subconcept of B and 50% confidence that it is a subconcept of $\neg B$. This is perfectly consistent under any reasonable interpretation of confidence values since a set can be equally similar or close to a pair of disjoint sets. However, the natural translation effectively strengthens the confidence which leads to an incoherence in the integrated ontology. We call such incoherences *spurious*.

The negative impact of spurious incoherences is that some mappings (or translations of mappings) need to be deleted in order to regain coherence. In the above example one will have to choose which axiom ($A \sqsubseteq B$ or $A \sqsubseteq C$) to delete. This leads to a loss of information and a potential loss of entailments from the integrated ontology.

It might be argued that a wise choice of a threshold solves the issue with spurious incoherences. In particular, the threshold can be set such that some of the mappings that cause incoherence do not get translated into axioms. However, this approach is not free of flaws either. First, it is not always clear how to pick the threshold, for example, the Conference Track results at OAEI-2009 suggest that different thresholds are sometimes used for different tools.[19] Second, a single threshold can discard too many mappings, for instance, in the above example any threshold will either leave the incoherence or discard both mappings. Third, *any* repair of incoherence would cause some loss of information because even the least confident mapping still represents a piece of knowledge which can potentially be exploited.

Probabilistic validation will be the most useful if the integrated ontology is going to be used as a probabilistic knowledge base. It does not necessarily mean that it should be used as a P-$\mathcal{SROIQ}$ ontology, for example, probabilities can be used at a meta-logical level for ranking query answers. Even in such cases, proving coherence will raise user confidence in correctness of (probabilistic) alignment tools since mappings can be explained via the well understood notion of probability distributions.

We plan to use our PSAT algorithm as a main validation tool. It can be used to check the integrated ontology for approximate probabilistic coherence, similarly to how

[19] See evaluation via reasoning at `http://oaei.ontologymatching.org/2009/results/conference/`

it has been done for CADIAG-2. There are two possible issues with this approach. The first is the general difficulty with approximate coherence since it is unclear how fine the approximation should be. The second concern is scalability because it may appear questionable whether the approach can be used for validating large alignments. Our aim is mostly to demonstrate that our algorithms and implementation make it feasible. To this end we have carried out an evaluation of the PSAT algorithm on probabilistic formalizations of large-scale alignments of the NCI Anatomy and the Mouse Anatomy ontologies from OAEI-2009. The challenges, results, and discussion are presented in Section 7.1.

Chapter 4

Understanding P-$\mathcal{SROIQ}$

This chapter presents two views on P-$\mathcal{SROIQ}$: as a generalization of propositional probabilistic logic of Nilsson [156, 157] (PPL) and as a fragment of Halpern and Bacchus' first-order logic of probability [76, 12] (FOPL_{II}). In the latter case we will mainly focus on monotonic properties of P-$\mathcal{SROIQ}$ and ignore the default reasoning mechanism, which can been inherited, with no substantial differences, from propositional logic with conditional constraints [133]. While the generalization presented in Section 4.1 is straightforward, the reduction of P-$\mathcal{SROIQ}$ to FOPL_{II} developed in Section 4.2 reveals some unobvious properties, in particular, what concerns probabilistic ABoxes. The implications of such properties on handling both statistical knowledge and degrees of belief are discussed in Section 4.3.

4.1 P-$\mathcal{SROIQ}$ as a Generalization of Propositional Probabilistic Logic

P-$\mathcal{SROIQ}$ was originally developed as a generalization of default propositional probabilistic logic [133] which, in turn, can be regarded as the basic Nilsson's PPL with one of the default reasoning mechanisms, e.g., lexicographic entailment by Lehmann [127]. We will illustrate the generalization only by translating PPL's knowledge bases into P-$\mathcal{SROIQ}$ and then discuss P-$\mathcal{SROIQ}$'s properties implied by the supported strength of entailment.

4.1.1 Basic Translation

Recall from Section 2.2.2 that a knowledge base in Nilsson's PPL, written as $\mathcal{K}_{PPL}$, is a set of probabilistic formulas which are pairs (s_i, p_i), where s_i is a propositional formula and p_i is a real number within $[0, 1]$. In this section we will call *PPL* a small extension of this basic model according to which a knowledge base $\mathcal{K}$ is a pair $(\mathcal{S}, \mathcal{P})$, where $\mathcal{S} = \{s_i\}$

is a finite set of propositional formulas (the classical part) and $\mathcal{P} = \{(r_i|s_i)[l_i, u_i]\}$ is a finite set of conditional probabilistic formulas with intervals (the probabilistic part). Its semantics is that of the Nilsson's logic except that probability distributions are defined over the set of truth assignments that satisfy all formulas in $\mathcal{S}$.[1] The notions required for lexicographic reasoning directly correspond to those of P-$\mathcal{SROIQ}$ [133].

We present the injective function κ which translates propositional knowledge bases into PTBoxes in P-$\mathcal{ALC}$. It uses two auxiliary functions, $\kappa_{PL\to\mathcal{ALC}}$ and $\kappa_{PPL\to P-\mathcal{ALC}}$, which translate the classical and the probabilistic parts respectively. We restrict the expressivity of classical part to $\mathcal{ALC}$ because it is a propositionally closed DL. The function translates each probabilistic formula into a constraint in P-$\mathcal{ALC}$:

$$\kappa((\{s_i\}, \{(r_i|s_i)[l_i, u_i]\})) = (\{\top \sqsubseteq \kappa_{PL\to\mathcal{ALC}}(s_i)\}, \{\kappa_{PPL\to P-\mathcal{ALC}}((r_i|s_i)[l_i, u_i])\}),$$
$$\kappa_{PPL\to P-\mathcal{ALC}}((r|s)[l, u]) = (\kappa_{PL\to\mathcal{ALC}}(r)|\kappa_{PL\to\mathcal{ALC}}(s))[l, u],$$

where $\kappa_{PL\to\mathcal{ALC}}$ translates each propositional formula, which is assumed to be in CNF, into a corresponding concept expression in $\mathcal{ALC}$:

$$\begin{aligned}
&\kappa_{PL\to\mathcal{ALC}}(a) = A_a,\\
&\kappa_{PL\to\mathcal{ALC}}(\neg s) = \neg\kappa_{PL\to\mathcal{ALC}}(s),\\
&\kappa_{PL\to\mathcal{ALC}}(s \wedge r) = \kappa_{PL\to\mathcal{ALC}}(s) \sqcap \kappa_{PL\to\mathcal{ALC}}(r),\\
&\kappa_{PL\to\mathcal{ALC}}(s \vee r) = \kappa_{PL\to\mathcal{ALC}}(s) \sqcup \kappa_{PL\to\mathcal{ALC}}(r),
\end{aligned}$$

where a, r, s, A denote a Boolean variable, propositional formulas, and a concept name respectively. It is easy to see that the translation preserves satisfiability and entailments:

Theorem 4.1. *Let $\mathcal{K}$ be a knowledge base in PPL and $PT = \kappa(\mathcal{K})$ be its translation. Then $\mathcal{K}$ is satisfiable whenever PT is satisfiable and $\mathcal{K} \models (r|s)[l, u]$ if and only if $PT \models (\kappa_{PL\to\mathcal{ALC}}(r)|\kappa_{PL\to\mathcal{ALC}}(s))[l, u]$.*

Proof. The result is an immediate consequence of the correspondence between models of $\mathcal{K}$ and PT. Let $\Phi_{PL} = \{a_1, \ldots, a_n\}$ be the set of Boolean variables appearing in $\mathcal{K}$ (the probabilistic signature). Then the corresponding signature of PT is simply $\Phi_{\mathcal{ALC}} = \{A_1, \ldots, A_n\}$. Let W_{PL} be a truth assignment to atoms in Φ_{PL}. The crucial fact is that it satisfies all formulas in $\mathcal{S}$ iff the concept expression $\bigsqcap_{i\in\{1,\ldots,n\}} B_i$, where B_i is equal to A_i if a_i is true in W_{PL} and $\neg A_i$ otherwise, is satisfiable w.r.t. $\mathcal{T}$ (because

[1] Conditional formulas are interpreted as in P-$\mathcal{SROIQ}$, i.e., an interpretation that assigns zero probability to a formula s vacuously satisfies all conditionals $(r|s)[l, u]$ for any r and $[l, u]$.

$\mathcal{ALC}$ faithfully subsumes propositional logic). Given the one-to-one correspondence between truth assignments in PPL and conjunctive concept expressions in P-$\mathcal{ALC}$ (i.e., possible worlds) we can define the obvious correspondence between probability distributions: $Pr_{P-\mathcal{ALC}}(W_{P-\mathcal{ALC}}) = Pr_{PL}(W_{PL})$. Then $Pr_{\mathcal{ALC}}((\kappa_{PL\to\mathcal{ALC}}(r)|\kappa_{PL\to\mathcal{ALC}}(s)))$ $= Pr_{PL}(r|s)$ for any propositional formula s and r over Φ_{PL}, so the result follows in both directions. □

The translation also preserves the results of lexicographic entailment because the notions of verification, falsification, toleration, and lexicographic ordering in P-$\mathcal{SROIQ}$ are exactly the same as in non-monotonic PPL presented in [133].

4.1.2 Strength of Entailments

In this section we discuss the strength of entailment supported by P-$\mathcal{SROIQ}$. PPLs can be classified with respect to strength of entailment for conditional formulas that they support [135]. Here we must distinguish between logical (monotonic) and lexicographic (non-monotonic) entailments. We begin with the former:

Definition 4.1 (λ-consequence)**.** *A formula $(r|s)[l,u]$ is a λ-**consequence** of $\mathcal{K}$, where $\lambda \in [0,1]$, denoted as $\mathcal{K} \models^{\lambda} (r|s)[l,u]$, if $\mathcal{K} \cup \{(r|\top)[\lambda,1]\} \models (r|s)[l,u]$. It is a tight λ-consequence, denoted as $\mathcal{K} \models^{\lambda}_{tight} (r|s)[l,u]$, if $\mathcal{K} \cup \{(r|\top)[\lambda,1]\} \models_{tight} (r|s)[l,u]$.*

Adapting the terminology from [135] we call logics which adopt 0-consequence (resp. 1-consequence) *monotonically weak* (resp. *monotonically strong*). Definitions of probabilistic consistency and lexicographic entailment from Section 2.3.2 can be similarly parametrized:

Definition 4.2 (λ-consistency)**.** *A probabilistic interpretation Pr λ-**verifies** a formula $(r|s)[l,u]$ if $Pr(s) = \lambda$ and $Pr(r) \in [l,u]$. Pr λ-**falsifies** $(r|s)[l,u]$ if $Pr(s) = \lambda$ and $Pr(r) \notin [l,u]$. A set of conditionals $\mathcal{F}$ λ-tolerates a conditional formula $(r|s)[l,u]$ under $\mathcal{K}$, if $\mathcal{K}$ has a model that λ-verifies $(r|s)[l,u]$. A KB $\mathcal{K}$ is λ-**consistent** if there exists an ordered partition $(\mathcal{K}_0,\ldots,\mathcal{K}_k)$ of $\mathcal{K}$, called z_λ-partition, such that each $\mathcal{K}_i$ $(i \in \{0,\ldots,k\})$ is the set of all conditional constraints from $\mathcal{K} \setminus \bigcup_{j=0}^{i-1} \mathcal{K}_j$ which are $\lambda-$tolerated by $\mathcal{P} \setminus \bigcup_{j=0}^{i-1} \mathcal{P}_j$ under $\mathcal{T}$.*

Finally, we need the parametrized notion of lexicographic entailment:

Definition 4.3 (λ-lex-preference and λ-lex-consequence)**.** *Given two probabilistic interpretations Pr_1, Pr_2 of $\mathcal{K}$ with z_λ-partition $(\mathcal{K}_0,\ldots,\mathcal{K}_k)$, Pr_1 is λ-**lex-preferable** to Pr_2 if there exists some $i \in \{0,\ldots,k\}$ such that $|\{\phi \in \mathcal{K}_i | Pr_1 \models \phi\}| > |\{\phi \in \mathcal{K}_i | Pr_2 \models \phi\}|$ and $|\{\phi \in \mathcal{K}_j | Pr_1 \models \phi\}| = |\{\phi \in \mathcal{K}_j | Pr_2 \models \phi\}|$ for all $i < j \leq k$. An interpretation* Pr *of $\mathcal{K}$ is λ-**lex-minimal** if no other interpretation of $\mathcal{K}$ is lex-preferable to it.*

A formula $(r|s)[l,u]$ *is a* λ-***lex-consequence*** *of* $\mathcal{K}$*, denoted as* $\mathcal{F} \models^{lex_\lambda} (r|s)[l,u]$*, if* $Pr(r) \in [l,u]$ *for every* λ*-lex-minimal model* Pr *of* $\mathcal{K}$ *that satisfies* $(s|\top)[\lambda,1]$*.* $(r|s)[l,u]$ *is a* ***tight*** λ***-lex-consequence*** *of* $\mathcal{K}$*, denoted as* $\mathcal{K} \models^{lex_\lambda}_{tight} (r|s)[l,u]$ *if* l *(resp.* u*) is the minimum (resp. the maximum) of* $Pr(r)$ *subject to all* λ*-lex-minimal models* Pr *of* $\mathcal{K}\{(s|\top)[1,1]\}$*.*

Analogously to the monotonic case, logics which adopt 0-lex-consequence (resp. 1-lex-consequence) are called *lexicographically weak* (resp. *lexicographically strong*). We ignore cases between the two extreme due to the lack of intuition behind them. The main reason why such a parametrization has been considered is the issue with *probabilistic inheritance* which is illustrated on the following example (adapted from [135]):

Example 4.1. *Consider the propositional knowledge base* $\mathcal{K} = (\{penguin \rightarrow bird\}, \{(see|yellow)[0.8,0.9], (fly|bird)[0.9,1], (fly|penguin)[0,0.1]\})$*, which informally states that "all penguins are birds", "generally, yellow objects are seen with probability between 0.8 and 0.9", "generally, birds fly with probability at least 0.9" , "generally, penguins fly with probability at most 0.1".*

Table 4.1 presents the results of weak and strong logical and lexicographic entailments from the KB in Example 4.1 for two conditional formulas (the probability that yellow birds are seen and the probability that yellow penguins fly). Regarding the first query, being easily seen is a probabilistic property of yellow objects. According to the bare theory of probability it is not necessarily inherited by a subclass of yellow objects, e.g., yellow birds. Indeed, yellow birds can be a very small class, e.g., one bird, which can either be easily seen or not, so the probability can freely vary between 1 and 0 respectively. This cautious, purely probabilistic intuition is supported by weak logical entailment. Strong logical entailment supports another intuition: since $\mathcal{K}$ contains no information about how easily yellow birds can be seen, one may assume that they behave just as other yellow objects, so the property should be *inherited* from the superclass (this is, in essence, reference class reasoning, see Section 2.2.2). In terms of default reasoning, strong entailment is braver by means of making implicit assumptions which may not be probabilistically sound. However, since it is a monotonic entailment and cannot resolve conflicts between probabilistic formulas, its strength can easily lead to failure to produce any interval. For example, it happens with the second query (the probability that yellow penguins fly) because $\mathcal{K} \cup \{(penguin|\top)[1,1]\}$ is unsatisfiable due to the pair of formulas $\{(fly|bird)[0.9,1]$,$(fly|penguin)[0,0.1]\}$.

Lexicographic entailment has been proposed to resolve such conflicts by *overriding*, i.e., preferring more specific knowledge (again, in the spirit of reference class reasoning). In the case of the second query it prefers the statement $(fly|penguin)[0,0.1]$ to

Table 4.1: Impact of the strength parameter on results of logical and lexicographic entailment (adapted from [135]). Recall that the result $[1, 0]$ means that the tightest probability interval is undefined due to inconsistency.

Query	Logical entailment		Lexicographic entailment	
	$\lambda = 0$	$\lambda = 1$	$\lambda = 0$	$\lambda = 1$
$(see\|yellow \wedge bird)$	[0,1]	[0.8,0.9]	[0,1]	[0.8,0.9]
$(fly\|yellow \wedge penguin)$	[0,1]	[1,0]	[0,1]	[0,0.1]

$(fly|bird)[0.9, 1]$. Then, after conflicts have been resolved, it uses logical entailment to compute the intervals. Here again, the strong notion sanctions inheritance from $(fly|penguin)[0, 0.1]$ while the weak one does not.

One unusual feature of P-$\mathcal{SROIQ}$, as developed by Lukasiewicz [136], is that it is monotonically weak but non-monotonically strong. In other words, TLogEnt is cautious while TLexEnt is brave. However, implementing weak TLogEnt is problematic because the result is entailed from all models which assign a *non-zero* probability to the evidence. This requires strict inequalities in the linear system (2.1) which cannot be handled by standard LP algorithms (this is related to the issue of coherence discussed in Section 2.3.4). Because of these practical considerations we have opted to support strong TLogEnt in P-$\mathcal{SROIQ}$ to make it consistent with TLexEnt.

As the previous example illustrated, strong TLogEnt can lead to inconsistencies which can be resolved via overriding. The question therefore becomes: can lexicographic ordering *always* "correct" the implications of such bravery to produce meaningful results? Unfortunately, the answer is "no" as we demonstrate for the following knowledge base (the example was first presented in [151]):

$$\mathcal{K} = (\emptyset, \{(automobile|car)[0.8, 0.9]\})$$

According to weak logical entailment this KB entails $(car|automobile)[0, 1]$ which, again, makes sense from the cautious point of view. On the other hand, strong TLogEnt produces the interval $[0, 0]$ for the same query, which seems surprising at first. The problem is that $\mathcal{K} \cup \{(automobile|\top)[1, 1]\}$ is incoherent, i.e., any model of it assigns zero probability to car, which of course implies that $Pr(car|X) = 0$, for any formula X. What is worse, this KB entails the same undesirable interval *lexicographically* because there are no conflicts to be resolved here, so the only lexicographically minimal models are the models of $\mathcal{K} \cup \{(automobile|\top)[1, 1]\}$. Furthermore, adding the formula $(car|automobile)[l, u]$ for any $0 < l \leq u < 1$ makes $\mathcal{K}$ *inconsistent* because the two formulas are now in conflict and none is preferable to the other, which completely

paralyzes lexicographic entailment.

This example is not completely artificial since the pair of conditional constraints $\{(C|D)[l,u],(D|C)[l,u]\}$ may look like a reasonable way to model that D is equivalent to C with probability between l and u. Such uncertain equivalences can come up when representing probabilistic ontology alignments. Instead, the user has to resort to one of the alternative ways of modeling equivalence, e.g., using the Jaccard coefficient (as we show in Section 3.3) or change the semantics of conditionals to make every satisfiable KB coherent and deal with the resulting computational difficulties.

4.2 P-$\mathcal{SROIQ}$ as a Fragment of First-Order Probabilistic Logic

This section presents a translation between P-$\mathcal{SROIQ}$ and the FOPL_{II}. For brevity we will limit our attention to $\mathcal{ALC}$ concepts (calling the resulting logic P-$\mathcal{ALC}$) as the translation can be extended to more expressive DLs in a straightforward, but technically involved way. We will show that the translation preserves entailments so that P-$\mathcal{ALC}$ (and, consequently, P-$\mathcal{SROIQ}$) can be viewed as a fragment of FOPL_{II}.

4.2.1 Translation of PTBoxes into FOPL

We define the injective function κ to be the mapping of P-$\mathcal{ALC}$ formulas to FOPL_{II}. It is a superset of the standard translation of $\mathcal{ALC}$ axioms into the formulas of FOL [18] (in the Table 4.2.1 A, B stand for concept names, R for role names, C, D for concept expressions, r for a fresh constant, $var \in \{x, y\}; var' = x$ if $var = y$ and y if $var = x$).

Table 4.2: Translation of P-$\mathcal{ALC}$ formulae into FOPL_{II}

P-$\mathcal{ALC}$	FOPL_{II}
$\kappa(A, var)$	$A(var)$
$\kappa(\neg C, var)$	$\neg(\kappa(C, var))$
$\kappa(R, var, var')$	$R(var, var')$
$\kappa(C \sqcap D, var)$	$\kappa(C, var) \wedge \kappa(D, var)$
$\kappa(C \sqcup D, var)$	$\kappa(C, var) \vee \kappa(D, var)$
$\kappa(\forall R.C, var)$	$\forall(var')(R(var, var') \rightarrow \kappa(C, var'))$
$\kappa(\exists R.C, var)$	$\exists(var')(R(var, var') \wedge \kappa(C, var'))$
$\kappa(a : C)$	$\kappa(C, x)[a/x]$
$\kappa((a, b) : R)$	$R(a, b)$
$\kappa(C \sqsubseteq D, x)$	$\forall(x)(\kappa(C, x) \rightarrow \kappa(D, x))$
$\kappa((D\|C)[l, u], x)$	$l \leq w(\kappa(D)(r)\|\kappa(C)(r)) \leq u$

For a possible world W we use the notation $\kappa(W)$ to denote the following conjunctive

formula with a single free variable: $\bigwedge\{\kappa(C)\}_{C\in W}\} \wedge \bigwedge\{\kappa(\neg C)\}_{C\notin W}\}$ (since each C is a concept, each $\kappa(C)$ is a monadic predicate).

This function transforms a P-$\mathcal{ALC}$ PTBox into a FOPL_{II} theory. The most important thing is that it translates *generic* PTBox constraints into *ground* probabilistic formulas for a fresh constant r, the same for all constraints. This explicates the fact that PTBox constraints are not (sort of) universally quantified statements which naturally apply to all probabilistic individuals but rather statements about a single object (the implications are discussed in the next section). Next we will show that this translation is faithful (i.e., it preserves satisfiability and entailments) and then generalize it to multiple PABoxes.

Faithfulness can be shown by establishing a correspondence between models in P-$\mathcal{ALC}$ and FOPL_{II}. Observe that in contrast to [116] we consider the natural choice of states in Type 2 probability structures in which they correspond to first-order models of the knowledge base.

Theorem 4.2. *Let $PT = (\mathcal{T}, \mathcal{P})$ be a PTBox in P-$\mathcal{ALC}$, where $\Sigma_\mathcal{T}$ and Φ stand for the signature of $\mathcal{T}$ and the probabilistic signature of PT respectively, and $F = \{\kappa(\phi)|\phi \in \mathcal{T} \cup \mathcal{P}\}$ be the translation according to Table 4.2.1. Then for every P-$\mathcal{ALC}$ model Pr of PT there exists a corresponding Type 2 structure $M = (D, S, \mu)$ such that:*

1. *for any axiom ϕ over $\Sigma_\mathcal{T}$, $(M, s) \models \kappa(\phi)$ for each $s \in S$ iff $\mathcal{T} \models \phi$,*
2. *for any Boolean concept expression X over Φ, $[w(\kappa(X)(r))]_M = Pr(X)$.*

and vice versa.

The first claim says that the translation preserves classical entailments over the signature of $\mathcal{T}$. The second claim implies that the translation preserves probabilities of concepts (they correspond to probabilities of ground formulas with the new constant r). The latter means that conditional probabilities are also preserved and, therefore, so are probabilistic entailments over Φ.

Proof. We first prove ($\Rightarrow$). Let $Pr : \mathcal{W}_\Phi \to [0, 1]$ be a model of PT (recall that $\mathcal{W}_\Phi$ is the set of all possible worlds over Φ, i.e., concept types which are realizable w.r.t. $\mathcal{T}$). Pr is a probability distribution so $\mathcal{W}_\Phi$ must be non-empty, which means that $\mathcal{T}$ is satisfiable (a concept type cannot be realizable w.r.t. an unsatisfiable TBox). We first define an extension of $\mathcal{T}$, called $\mathcal{T}'$, as follows: $\mathcal{T}' = \mathcal{T} \cup \bigcup_{W\in\mathcal{W}_\Phi}\{W(o_w)|W \in \mathcal{W}_\Phi\}$, where $W(o_w) = \{\{o_w : C\}_{C\in W} \cup \{\neg o_w : C\}_{C\notin W}\}$ and for each world o_w is a new individual name.[2] $\mathcal{T}'$ has exactly the same set of entailments w.r.t. $\Sigma_\mathcal{T}$ as $\mathcal{T}$ (we

[2]So in this case $\mathcal{T}'$ is a combination of a TBox and an ABox. In contrast to $\mathcal{SROIQ}$, assertions of the form $C(w)$ are not expressible via TBox axioms in $\mathcal{ALC}$.

simply ruled out all models of $\mathcal{T}$ which do not realize some world[3]). Therefore, it is sufficient to prove the claims w.r.t. $\mathcal{T}'$.

Now we use $\Im$ for the set of all models of $\mathcal{T}'$ and define $M = (D, S, \mu)$ as follows (the details are presented below):

- $D = \bigcup\{\Delta^{\mathcal{I}} | \mathcal{I} \in \Im\}$,
- $s(\mathcal{I}, \cdot, Z) = \kappa^{-1}(Z)^{\mathcal{I}}$, if Z is a translation of a concept or a role over $\Sigma_{\mathcal{T}}$,
- $s(\mathcal{I}, W, r) = d \in \Delta^{\mathcal{I}}$, for such d that $d \in C^{\mathcal{I}}$, if $C \in W$, and $d \in (\neg C)^{\mathcal{I}}$ otherwise,
- $S = \{s(\mathcal{I}, W, \cdot) | \mathcal{I} \in \Im, W \in \mathcal{W}_{\Phi}\}$,
- $\mu(\sigma(W)) = Pr(W)$ for each $W \in \mathcal{W}_{\Phi}$, where $\sigma(W) = \{s(I, W, \cdot) \in S | s(I, W, r) \in s(I, \cdot, \kappa(W))\}$.

We take the domain D to be the domain union for all models of $\mathcal{T}'$. Next we define the interpretation function $s(\mathcal{I}, W, \cdot)$ which interprets each predicate Z (i.e., a translation of either a concept or a role) in the same way as $\cdot^{\mathcal{I}}$ interprets $\kappa^{-1}(Z)$. In addition, it interprets the new constant r as some realization of the world W in $\mathcal{I}$ (here we use the fact that all worlds are realized in models of $\mathcal{T}'$). Then we take the set of states as all possible interpretations s over $\Im$ and $\mathcal{W}_{\Phi}$. On the last step we define the probability distribution μ over S. For that we first define a function σ which maps each world $W = \{C_1, \ldots, C_k\}$ to a subset of states $\sigma(W) \subseteq S$ as follows: $\sigma(W) = \{s \in S | s \models \kappa(W)(r)\}$ ($s \models \kappa(W)(r)$ is equivalent to $s(I, W, r) \in s(I, \cdot, \kappa(W))$. Intuitively, $\sigma(W)$ is a set of first-order interpretations which satisfy $\kappa(C_i)(r)$ iff $C_i \in W$, so there is a one-to-one correspondence between worlds on the P-$\mathcal{ALC}$ side and interpretations of r on the FOPL$_{\text{II}}$ side. Finally, we take $\mu(\sigma(W)) = Pr(W)$ for each possible world.

The first claim follows from the second bullet above because κ encompasses a standard and faithful translation from $\mathcal{ALC}$ to FOL and r is a fresh constant (so can be ignored for entailments over $\Sigma_{\mathcal{T}}$ which does not include it). The second claim is more complicated. First, observe that μ is a probability distribution (i.e. non-negative and countably additive) because it mimics the probability distribution Pr. The probability of a concept expression X over Φ, i.e., $Pr(X)$, is defined as $\sum_{W \models X} Pr(W)$, which is equal to $\sum_{W \models X} \mu(\sigma(W))$ or, using the definition of σ, equals to $\sum_{W \models X} \mu(\{s | s \models \kappa(W)(r)\})$, which is exactly $\mu(\{s | s \models \kappa(X)(r)\})$ or $[w(\kappa(X)(r))]_M$.

We now sketch the proof of ($\Leftarrow$). Let $M = (D, S, \mu)$ be a Type II model of F. We construct an $\mathcal{ALC}$ interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ as follows: $\Delta^{\mathcal{I}} = D$ and $(\kappa^{-1}(\phi))^{\mathcal{I}} = s(\phi)$ for an arbitrary $s \in S$ (since all classical formulas ϕ are satisfied in every state). $\mathcal{I} \models \phi$ due to the faithfulness of the $\mathcal{ALC}$ to FOL translation, so the first claim holds.

[3]This step will be more awkward for logics with nominals, e.g., $\mathcal{SROIQ}$, since they can force an upper bound on the cardinality of the domain.

To construct Pr we first construct the set of possible worlds $\mathcal{W}_\Phi$. We take $\Phi = \bigcup_{(l \leq w(\psi(r)|\phi(r)) \leq u) \in F} \{\kappa^{-1}(\psi), \kappa^{-1}(\phi)\}$ as the probabilistic signature of PT. The key is that for every world $W \subseteq \Phi$, it satisfies $\mathcal{T}$ iff $\kappa(W)(r)$ is satisfiable w.r.t. F (since $\Delta^{\mathcal{I}} = D$, W could be a concept type of $s(r)$ for some s which satisfies $\kappa(W)(r)$). Let $\mathcal{W}_\Phi$ be the set of worlds which satisfy $\mathcal{T}$. Now we take $Pr(W) = \mu(\sigma(W))$, where $\sigma(W) = \{s \in S \mid s \models \kappa(W)(r)\}$. The second claim now follows in the same way is in ($\Rightarrow$). □

The following result is a straightforward corollary of the above theorem.

Corollary 4.1. *Let $PT = (\mathcal{T}, \mathcal{P})$ be a PTBox in P-$\mathcal{ALC}$, where $\Sigma_\mathcal{T}$ and Φ stand for the signature of $\mathcal{T}$ and the probabilistic signature of PT respectively, and $F = \{\kappa(\phi) | \phi \in \mathcal{T} \cup \mathcal{P}\}$ be the translation according to Table 4.2.1. Then the following is true:*

1. *PT is satisfiable iff F is satisfiable,*
2. *$PT \models (D|C)[l, u]$ iff $F \models l \leq w(\kappa(D)(r)|\kappa(C)(r)) \leq u$.*

In the next section we extend the translation from PTBoxes to probabilistic KBs.

4.2.2 Translation of PABoxes into FOPL

One particularly odd characteristic of P-$\mathcal{SROIQ}$ is that PABoxes cannot be combined into a single set of formulas. The separation between PABoxes and the PTBox can partly be justified because they are meant to contain different kinds of probabilistic knowledge, i.e., generic relationships and information about particular individuals respectively. However, the same argument does not hold in the case of separated PABoxes. Their separation has purely technical foundations: PABox constraints are modeled as generic constraints and the information about the individual is present only on a meta-level (as a label of the PABox). Therefore, to extend our translation to PABoxes we either have to translate them into a corresponding disjoint set of FOPL_{II} theories (with similar meta-labels) or make special arrangements to faithfully translate them into a combined FOPL_{II} theory. We opt for the latter because it will let us get rid of any meta-logical aspects and view a P-$\mathcal{SROIQ}$ ontology as a single, standard theory in FOPL_{II}.

Since PABoxes in P-$\mathcal{SROIQ}$ are isolated from each other, the translation should preserve that isolation. The most obvious way to prevent any interaction between sets of formulas in a single logical theory is to make their signatures disjoint. That is, PABoxes can be translated into FOPL_{II} sub-theories with disjoint signatures. However, the translation should not only respect disjointness of PABoxes but also preserve their interaction with PTBox and the classical part of the ontology. We give an example to illustrate the issue.

Example 4.2. *Consider the following PTBox:* $PT = \{\emptyset, \{(FlyingObject|Bird)[0.9, 1], (FlyingObject|\neg Bird)[0, 0.5]\}$ *and two PABoxes:* $\mathcal{P}_{Tweety} = \{(Bird|\top)[1, 1]\}, \mathcal{P}_{Sam} = \{(\neg Bird|\top)[1, 1]\}$. *If these sets of axioms are translated and combined into a single* FOPL_{II} *theory then it will contain a conflicting pair of formulas* $\{w(Bird(r)) \geq 0.9, w(Bird(r)) \leq 0.5\} \subseteq F$.

This inconsistency can be avoided by introducing fresh first-order predicates for every PABox: $\{w(Bird_{Tweety}(r)) \geq 0.9,\ w(Bird_{Sam}(r)) \leq 0.5\}$. *However, this would break any connection between PTBox and PABox axioms, for example, prevent the entailments* $\{w(FlyingObject_{Tweety}(r)) \geq 0.9,\ w(FlyingObject_{Sam}(r)) \leq 0.5\}$.

Another way to faithfully extend the translation to PABoxes is to introduce fresh concept names to *relativize* each TBox and PTBox axiom for every probabilistic individual and thus avoid inconsistencies. More formally, the transformation consists of the following steps:

- Firstly, we transform a P-$\mathcal{ALC}$ ontology $PO = (\mathcal{T}, \mathcal{P}, \{\mathcal{P}_o\}_{o \in N_{PI}})$ into a set of PT-Boxes $\{(\mathcal{T}, \mathcal{P})\} \cup \{(\mathcal{T}, \mathcal{P} \cup \mathcal{P}_o)\}_{o \in N_{PI}}$. Informally, we create a copy PTBox for every probabilistic individual (PT_o) and make them isolated from each other. Now, instead of one PTBox and a set of PABoxes we have just a set of PTBoxes. This step preserves probabilistic entailments in the following sense: $PO \models (B|A)[l, u]$ iff $(\mathcal{T}, \mathcal{P}) \models (B|A)[l, u]$ and $PO \models (B|A)[l, u]$ for o iff $PT_o \models (B|A)[l, u]$ (classical entailments are trivially preserved).

- Secondly, we transform every PTBox PT_o into PT'_o by renaming every concept name C into C_o in all TBox axioms and conditional constraints. It is easy to see that $PT_o \models C \sqsubseteq D$ iff $PT'_o \models C_o \sqsubseteq D_o$ and $PT_o \models (B|A)[l, u]$ iff $PT'_o \models (B_o|A_o)[l, u]$. Intuitively, we have created a fresh copy of each PTBox to guard against possible conflicts between PABox constraints for different probabilistic individuals. Signatures of PT'_o are pairwise disjoint and denoted as Σ_o.

- Next, we union all PT'_o with disjoint signatures (including the original $PT = (\mathcal{T}, \mathcal{P})$) into a single unified PTBox $PT_U = \bigcup_{o \in I_p} PT_o \cup PT$ with signature $\Sigma_U = \bigcup_{o \in I_p} \Sigma_o \cup \Sigma$.

- Finally we can apply the previously presented faithful translation to PT_U and obtain a single FOPL_{II} theory which corresponds to the original P-$\mathcal{ALC}$ ontology.

A necessary condition for faithfulness of this transformation is that the original isolation of PABoxes is preserved by creating fresh copies of PTBoxes. In particular, this means that the unified PTBox cannot entail any subsumption relation between concept expressions C_{o_1} and C_{o_2} defined over disjoint signatures except of the case

when one of them is either $\top$ or $\bot$. If this is false, for example, if $PT_U \models C_{o_1} \sqsubseteq C_{o_2}$ then the following PABox constraints represented as $(C_{o_1}|\top)[1,1]$ and $(C_{o_2}|\top)[0,0]$ will be contradictory in PT_U (but they were consistent in the original P-$\mathcal{ALC}$ because they belonged to different PABoxes isolated from each other). This condition is formalized in the following lemma:

Lemma 4.1. *Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be copies of a satisfiable $\mathcal{ALC}$ ontology $\mathcal{T}$ with disjoint signatures Σ_1 and Σ_2, and $\mathcal{T}_U$ be the union of $\mathcal{T}_1$ and $\mathcal{T}_2$. Then for any concept expressions C_1, C_2 over Σ_1 and Σ_2 respectively such that $\mathcal{T}_1 \not\models C_1 \sqsubseteq \bot$ and $\mathcal{T}_1 \not\models \top \sqsubseteq C_2$, $\mathcal{T}_U \not\models C_1 \sqsubseteq C_2$.*

Proof. Let $\mathcal{I}_1 = (\Delta^{\mathcal{I}_1}, \cdot^{\mathcal{I}_1})$ and $\mathcal{I}_2 = (\Delta^{\mathcal{I}_2}, \cdot^{\mathcal{I}_2})$ be models of $\mathcal{T}_1$ and $\mathcal{T}_2$ respectively, $x \in C_1^{\mathcal{I}_1}, y \in \Delta^{\mathcal{I}_2} \setminus C_2^{\mathcal{I}_2}$. We can assume that $\Delta^{\mathcal{I}_1}$ and $\Delta^{\mathcal{I}_2}$ are countably infinite because $\mathcal{ALC}$ models are closed under disjoint union. Next, we choose two linear orderings $p_i : \Delta^{\mathcal{I}_i} \to \mathbb{N}$ $(i \in \{1,2\})$ such that $p_1(x) = p_2(y) = 1$ and pick a new countable domain $\Delta^{\mathcal{I}_U} = \{d_{U_1}, d_{U_2}, \dots\}$. Finally, we construct an interpretation function $\cdot^{\mathcal{I}_U}$ such that for any concept name C_i (resp. role name R_i), $C_i^{\mathcal{I}_U} = \{d_{U_j} | p_i^{-1}(j) \in C_i^{\mathcal{I}_i}\}$ (resp. $R_i^{\mathcal{I}_U} = \{(d_{U_j}, d_{U_k}) | (p_i^{-1}(j), p_i^{-1}(k)) \in R_i^{\mathcal{I}_i}\}$).

Informally, we order both domains such that x and y are in the first position each. Then the domains are *aligned* such that elements at the same position, e.g., x and y, coincide. This induces a model $\mathcal{I}_U = (\Delta^{\mathcal{I}_U}, \cdot^{\mathcal{I}_U})$ of $\mathcal{T}_U$ which interpretation function agrees with $\mathcal{I}_i$ on all concepts and roles from Σ_i and which does *not* satisfy $C_1 \sqsubseteq C_2$. □

Now we can obtain the main result:

Theorem 4.3. *Let $PO = (\mathcal{T}, \mathcal{P}, \{\mathcal{P}_o\}_{o \in N_{PI}})$ be a P-$\mathcal{ALC}$ ontology, where $\Sigma_\mathcal{T}$ and Φ_o stand for the signature of $\mathcal{T}$ and the probabilistic signature of $(\mathcal{T}, \mathcal{P} \cup \mathcal{P}_o)$ for $o \in N_{PI}$ respectively. Let F be the* $\mathrm{FOPL_{II}}$ *theory obtained by combining the PABoxes and translating the resulting PTBox into* $\mathrm{FOPL_{II}}$*. If all PTBoxes of the form $PT_o = (\mathcal{T}, \mathcal{P} \cup \mathcal{P}_o)$ are satisfiable, then for every P-$\mathcal{ALC}$ model Pr_o of every PT_o there exists a corresponding Type 2 structure $M = (D, S, \mu)$ such that:*

1. *for any axiom ϕ over $\Sigma_\mathcal{T}$, $(M, s) \models \kappa(\phi)$ for each $s \in S$ iff $\mathcal{T} \models \phi$,*
2. *for any Boolean concept expression X over Φ_o, $[w(\kappa(X)(r))]_M = Pr(X)$.*

and vice versa. Furthermore, $\mathcal{F}$ is unsatisfiable iff $(\mathcal{T}, \mathcal{P} \cup \mathcal{P}_o)$ is unsatisfiable for some $o \in N_{PI}$.

Proof. Due to Theorem 4.2 it suffices to show that the steps 1-3 of the transformation preserve probabilistic models. This can be done by establishing a correspondence between possible worlds of each PT_o and PT_U. Since there are no subsumptions between

concept expressions over signatures of different PTBoxes (see Lemma 4.1), each possible world W_o in PT_o corresponds to a finite set of possible worlds of PT_U defined as: $\sigma(W_o) = \{W_U \mid C_{i_o} \in W_U \text{ iff } C_i \in W_o\}$ (each C_{i_o} is a new concept name for C_i introduced on step 2). Then, a probability distribution over all possible worlds in PT_U can be defined as $Pr_U(W_U) = Pr_o(W_o)/|\sigma(W_o)|$. It follows that for any concept C over Σ_o, $Pr_o(C)$ is equal to $Pr_U(C_o)$ where C_o is the correspondingly renamed concept. Therefore, $Pr_U \models (B_o|A_o)[l,u]$ if $Pr_o \models (B|A)[l,u]$. The reverse direction can be proved along the same lines (i.e., $Pr_o(W_o)$ can be defined as $\sum_{W_U \in \sigma(W_o)} Pr_U(W_U)$). □

It is worth stressing that if PABox for some probabilistic individual o contradicts the PTBox $(\mathcal{T}, \mathcal{P})$ then the entire FOPL_{II} theory is unsatisfiable. Therefore, for practical considerations it might be important to work with P-$\mathcal{SROIQ}$ ontologies as with stratified theories. Such separation between general knowledge and knowledge about particular individuals had been known before P-$\mathcal{SROIQ}$, for example, it was used in the default reasoning system developed by Geffner and Pearl [61].

4.3 Properties and Limitations of P-$\mathcal{SROIQ}$

The translation highlights two major properties of P-$\mathcal{SROIQ}$:

PI P-$\mathcal{SROIQ}$ has a subjective, interpretation-based semantics.

PII Only a single constant is required to translate all probabilistic knowledge in a P-$\mathcal{SROIQ}$ ontology into a FOPL_{II} theory.[4]

PI implies that any claims that P-$\mathcal{SROIQ}$ handles different kinds of probabilities, especially statistics, require a careful examination. PII, which is the basis of P-$\mathcal{SROIQ}$'s direct inference mechanism, explains issues with handling degrees of belief since, intuitively, a single constant cannot be sufficient for modeling probability distributions over relational structures. We will discuss these issues in 4.3.2 and 4.3.3 but before we briefly discuss why some other, perhaps more naturally looking ways of translating P-$\mathcal{SROIQ}$ into FOPL_{II} are incorrect.

4.3.1 Interpretation of Probabilistic Statements

According to the translation, all probabilistic statements in P-$\mathcal{SROIQ}$ express *degrees of belief* about a single, yet unnamed, individual (denoted as r). This is not an easily expected outcome because the variable-free syntax may give a misleading impression

[4]Here we mean a "probabilistic" constant since all nominals occurring in the *classical* part of the ontology will be translated into corresponding constants in FOPL_{II}.

that PTBox constraints correspond to universally quantified formulas in FOPL_{II}, similarly to how TBox axioms in $\mathcal{SROIQ}$ correspond to universally quantified implications formulas in FOL. One may wonder whether a more natural translation is possible. We consider two such candidate translations: probabilistic implications and universally quantified conditional formulas.

An interpretation of conditional constraints $(D|C)[l,u]$ as formulas of the form $l \leq w(\forall x[c(x) \rightarrow d(x)]) \leq u$ lets us view them as probabilistic generalizations of TBox axioms $C \sqsubseteq D$ (which are translated into $\forall x[c(x) \rightarrow d(x)]$). It is easy to see how their semantics is different from P-$\mathcal{SROIQ}$'s. Such formulas are unconditional so, for example, the pair of formulas $w(\forall x[c_1(x) \rightarrow d(x)]) \geq 0.9$ and $w(\forall x[c_1(x) \wedge c_2(x) \rightarrow d(x)]) \leq 0.8$ are contradictory. On the other hand, the pair of conditional constraints in P-$\mathcal{SROIQ}$ $(D|C_1)[0.9,1]$ and $(D|C_1 \sqcap C_2)[0,0.8]$ is perfectly satisfiable.[5]

The translation into universally quantified *conditional* formulas, i.e., formulas of the form $\forall x[l \leq w(d(x)|c(x)) \leq u]$ has more subtle issues. The idea of using them for capturing statistical assertions is originally due to Cheeseman [31]. It has been criticized by multiple authors (see esp. [76, 11, 12]) as it leads to intuitively unreasonable conflicts between statistics and beliefs. We will return to this point in the next section while here we can show that such translation is unfaithful in presence of named constants (i.e., nominals in $\mathcal{SROIQ}$) or classical ABoxes. For example, the PTBox $(\{a : \neg A\}, \{(A|\top)[1,1]\})$ is satisfiable in P-$\mathcal{SROIQ}$ although the corresponding FOPL_{II} theory $\{\neg A(a), \forall x(w(A(x)) = 1)\}$ is not. The problem is that this translation disregards the separation between classical and probabilistic individuals in P-$\mathcal{SROIQ}$.

In fact, the translation into quantified statements *does* work but requires a somewhat non-standard quantifier. It has to make bound variables act as *random designators*. This is precisely what we achieve by using the fresh constant r.

4.3.2 Representation of Statistics

The first question that has to be raised is whether P-$\mathcal{SROIQ}$ can be used to represent statistical knowledge given its subjective, interpretation based semantics. Here we prefer to distinguish between *practical* and *philosophical* difficulties. The former are the situations when some important statistical knowledge cannot be adequately represented, for example, all possible representations lead to statistically unsound conclusions. The latter are the situations which cause conceptual difficulties but do not lead to any erroneous entailments.

The main *philosophical* difficulty in P-$\mathcal{SROIQ}$ is that it enforces the separation between general statements, which are meant to capture statistics, and statements

[5] In other words, conditional formulas do not constrain future beliefs after conditioning on new evidence. This is related to the lack of probabilistic inheritance from the cautious point of view, see the previous discussion in Section 4.1.2.

meant to represent beliefs about specific individuals. This is a well-known argument against representing both statistics and beliefs in $\mathrm{FOPL}_{\mathrm{II}}$ (see [12, 76]). Consider the following classical example:

$$\begin{aligned} PO = (&\{Penguin \sqsubseteq Bird\}, \\ &\{(FlyingObject|Bird)[0.9, 1], (FlyingObject|Penguin)[0, 0.1]\}, \\ &(\{(Penguin|\top)[1, 1]\}_{Tweety})) \end{aligned}$$

If all axioms above were combined in a single theory it would clearly be unsatisfiable. The TBox and PTBox axioms place restrictions on probability of *Penguin* (informally, penguins must be a "small" subclass of birds) which is violated by the PABox axiom. This means that an agent cannot *simultaneously* believe in the existence of a single flying penguin and the statistical knowledge that most penguins do not fly, which is unreasonable. Since there is no semantic separation (i.e., through different probability distributions as in $\mathrm{FOPL}_{\mathrm{III}}$) between different kinds of axioms, they have to be separated syntactically. In addition, P-$\mathcal{SROIQ}$ has to include a special mechanism for combining these axioms for reasoning about individuals which has to be non-monotonic.

Interestingly, P-$\mathcal{SROIQ}$, as it stands, seems to avoid *practical* issues with handling statistics, but mostly because its language is quite limited rather than because its semantics is appropriate. The only probabilistic axioms provided by P-$\mathcal{SROIQ}$, conditional constraints of the form $(D|C)[l, u]$, express that "the probability that a random instance of C is an instance of D is in $[l, u]$". It does not allow specifying *how* that random instance was drawn as well as placing any other restrictions on probability distributions. It is easy to show that possible extensions in these directions could easily reveal the inadequacy of P-$\mathcal{SROIQ}$'s semantics for handling statistics.

Consider what happens if one wants to extend P-$\mathcal{SROIQ}$ to allow restricting probability functions to uniform distributions. This is useful if conditional constraints are to be interpreted as proportions (i.e., according to the frequentist interpretation of probability). Now consider the following PTBox where *marriedTo* is a functional role:

$$\begin{aligned} (&\{Person \sqsubseteq Man \sqcup Woman, Man \sqcap Woman \sqsubseteq \bot, Man \sqsubseteq \exists marriedTo.Woman\}, \\ &\{(Person|\top)[0.9, 0.9], (Man|Person)[0.5, 0.5]\}) \end{aligned}$$

This PTBox attempts to model a domain 90% of which consists of people. Every person is either a man or a woman. Furthermore, 50% of people are men and every man is *functionally* related to at least one woman, so the other half of people must be women. Due to the standard P-$\mathcal{SROIQ}$ semantics the PTBox will entail $[0, 1]$ as the tightest

probability bounds for the query $(Woman|Person)[?, ?]$. This happens because the relationship between *extensions* of Man and $Woman$ is ignored by the semantics of P-$\mathcal{SROIQ}$, i.e., it does not restrict the set of possible worlds in any way. Restricting probability functions to uniform distributions over possible worlds without changing the notion of possible world also does not achieve the goal (in this example there are fewer than 10 possible worlds, so constraints like $(Person|\top)[0.9, 0.9]$ would be unsatisfiable by themselves). Such considerations lead us to the conclusion that P-$\mathcal{SROIQ}$ is not well suited for representing *first-order* statistical statements.

4.3.3 Representation of Beliefs

Perhaps surprisingly the properties of P-$\mathcal{SROIQ}$, in particular, PII, lead to more practical difficulties with handling degrees of belief rather than representation of statistics. The prime issues are the separation between classical and probabilistic individuals and the lack of relational structures support.

The separation between different kinds of individuals precludes any combination between classical and probabilistic knowledge for the same individual. It is not possible, for example, to express that Mary is an instance of concept $Woman$ and has 90% chance of having BRCA1 gene mutation. Of course, it is possible to express that *probability* that Mary is a woman is 1 but this is not always a satisfactory replacement for ABox statements. First, one may want to specify probabilistic facts about individuals already present in the ABox. Second, perhaps more importantly, specifying ABox axioms as PABox axioms does not lead to entailments which could be important. For example, if Mary is a woman and developed breast cancer, then her daughter, say, Jane, would be entailed as an instance of concept $WomanWithFamilyHistoryOfBRCA$. If Mary has to be a probabilistic individual then so does Jane, and the modeler will face the problem of representing their relationship, which, in fact, is the second issue with P-$\mathcal{SROIQ}$.

P-$\mathcal{SROIQ}$ does not support probabilistic relational structures in the sense that one cannot specify that one probabilistic individual has a certain probability of being related to another probabilistic individual. For example, if both Mary and Jane are probabilistic individuals one cannot specify that Mary is a mother of Jane with a probability of 1 (obviously, such a statement is most reasonably represented as a classical ABox axioms but this is also not possible due to the separation discussed above).[6] The reason, which is highlighted by PII, is that PABox statements do *not* correspond to ground probabilistic formulas in FOPL_{II}. The information about individuals is present only on a meta-level, as labels of the corresponding PABoxes. As a consequence, knowledge about distinct probabilistic individuals has to be separated from each other, for

[6] Note that the logic can represent probabilistic roles between a classical and a probabilistic individuals, e.g., as a PABox axiom $(\exists motherOf.\{Jane\}|\top)[1, 1]$ for $Mary$, but in this case no probabilistic facts can be specified for Jane.

example, by means of isolated PABoxes in P-$\mathcal{SROIQ}$ or disjoint signatures in FOPL_{II}.

The second issue appears to be more difficult to overcome than the first. The separation between classical and probabilistic individuals can be eliminated by incorporating classical individuals in the description of possible worlds, for example, by including nominal concept expressions in the set of basic concepts (the probabilistic signature). Relational structures, on the other hand, cannot be supported until PABox constraints are interpreted as PTBox constraints rather than ground statements bearing information about particular individuals on the *logical level*. However, that would require major semantic changes, at least a new direct inference mechanism to preserve interaction between PTBox and PABox knowledge (a discussion of such possibility can be found in [113]).

4.3.4 Summary

We presented a faithful translation of knowledge bases in P-$\mathcal{SROIQ}$ into theories in first-order probabilistic logic with Type II semantics. The translation places no restriction on expressiveness (e.g., use of nominals), uses only standard quantifiers and, most importantly, illuminates the probabilistic propositionality of P-$\mathcal{SROIQ}$ by using only a single probabilistic constant. That "propositionality" is the main culprit of the important limitations of P-$\mathcal{SROIQ}$, namely, the lack of probabilistic relational structures.

P-$\mathcal{SROIQ}$ can be seen as an *approximation* of FOPL_{III} which trades separate probability distributions for statistical and belief statements for a practically implementable direct inference mechanism. There is nothing particularly wrong with such a design decision *per se* but it has to be understood by modelers. Our translation into FOPL_{II} is an attempt to enhance that understanding analogously to how classical DLs are understood as fragments of classical FOL.

In conclusion, we present two examples that illustrate how viewing probabilistic DLs as fragments of FOPL helps their understanding and, on the contrary, how lack of such understanding can lead to errors. An example of the latter is the design of P-$\mathcal{SHOQ}$(D), a predecessor of P-$\mathcal{SROIQ}$ [65], which has domain-based (Type I) semantics. In that logic PABox axioms are represented using nominals, for example, $(C|\{a\})[0.5, 1]$ is supposed to model that a is an instance of C with probability at least 0.5. However, as proved by Halpern, closed first-order formulas can only have probability 0 or 1 in any Type 1 probabilistic model (see Lemma 2.3 in [76]) so the representation is unsatisfactory. It is easy to see that the probability of $(C|\{a\})$, which is equivalent to $\frac{Pr(C \sqcap \{a\})}{Pr(\{a\})}$, is 0 if $a^{\mathcal{I}} \notin C^{\mathcal{I}}$ or 1 if $a^{\mathcal{I}} \in C^{\mathcal{I}}$, if Pr is a probability distribution over $\Delta^{\mathcal{I}}$.

A positive example is the recent work of Lutz and Schröder [139]. They designed and *presented* a probabilistic DL, called Prob-$\mathcal{ALC}$, which supports probabilistic concepts

of the form $P_{\geq\alpha}(C)$, which are interpreted as "the set of all domain objects which are instances of C with $\geq 90\%$ probability". The logic is designed and presented as a fragment of FOPL_{II} so a modeler can immediately realize that their ability to model statistical statements might be limited. And it is indeed the case, for example, if one tries to use axioms like $Bird \sqsubseteq P_{\geq 0.9}(FlyingObject)$ to capture the statistical statement that $\geq 90\%$ of birds fly, they will face the same difficulties as with using universally quantified probabilistic formulas in FOPL_{II}. In particular, such an axiom will be in conflict with statements asserting existence of a *specific* non-flying bird, i.e., $\{tweety : Bird, tweety : \neg FlyingObject\}$. On the other hand, in contrast to P-$\mathcal{SROIQ}$, the logic fully supports relational structures in probabilistic ABoxes, something that could be expected from a FOPL_{II} frament.

Chapter 5

Algorithms for Practical Reasoning in P-$\mathcal{SROIQ}$

This chapter describes the algorithms for the main reasoning tasks in P-$\mathcal{SROIQ}$ that have been developed in this thesis. Unlike the original algorithms described in Chapter 2 these procedures aim at being practical in realistic scenarios involving reasonable amounts of probabilistic knowledge, i.e. hundreds of conditional constraints. The core component—the PSAT algorithm based on column generation—is described in Section 5.1 which also presents its optimizations, analysis, and compares to the previous approaches. Sections 5.2, 5.3, and 5.4 then describe other reasoning procedures for analysis of probabilistic knowledge bases, deciding probabilistic consistencies, and, finally, computing tight lexicographic entailment.

5.1 The Probabilistic Satisfiability Algorithm

The principal contribution of this thesis is the novel PSAT algorithm implemented in Pronto (see Section 5.1.6 for comparison with the previously developed methods). For the sake of clarity we will consider a special case of PSAT where the PTBox is of the form $PT = (\mathcal{T}, \{(C_i|\top)[p_i, p_i]\})$ (i.e. all probabilistic statements are unconditional constraints with point-valued probabilities and all C_i are concept names). It is straightforward, but technically awkward, to generalize the procedure to handle conditional interval statements over arbitrary concept expressions. Also, essentially the same algorithm can be applied to solve TLogEnt problem with the only difference that the linear program is optimized twice to get the lower and the upper probabilities.

A PTBox $PT = (\mathcal{T}, \{(C_i|\top)[p_i, p_i]\})$ is satisfiable iff the following system of linear inequalities is *feasible*, i.e. admits at least one solution (by generalization from propositional PSAT [81]):

$$\sum_{W \models C_i} x_W = p_i, \text{ for each } (C_i|\top)[p_i, p_i] \in \mathcal{P} \qquad (5.1)$$
$$\sum_{W \in \mathcal{W}_\Phi} x_W = 1 \text{ and all } x_W \geq 0$$

where $\mathcal{W}_\Phi$ is the set of all possible worlds for the set of concepts Φ in $\mathcal{T}$. Observe, that $\mathcal{W}_\Phi$ is finite and exponential in the size of Φ so it is not practical to try to explicitly generate this system in order to check whether it has a solution.

One successful approach to dealing with linear systems having an exponential number variables is *column generation* [42]. It is based on the fundamental property of linear programming: any feasible program (i.e., a program that admits at least one solution) always has an optimal solution in which only a linear number of variables have non-zero values. Column generation exploits this property by trying to avoid an explicit representation of variables (columns) which will not have positive values in the finally discovered solution. The method is outlined in the next subsection.

5.1.1 Column Generation Basics

Consider the standard form of a linear program (5.2). Any linear program, in particular, a version of (5.1) with intervals can be reduced to it by adding slack variables.

$$\begin{aligned} \max\ & z = cx \\ \text{s.t. } & Ax = b \\ & x \geq 0 \end{aligned} \qquad (5.2)$$

A denotes a $m \times n$ matrix of linear coefficients of (5.2). At every step of the simplex algorithm,[1] A is represented as a combination (B, N) where B and N are submatrices of the *basic* and *non-basic* variables, respectively. Values of non-basic variables are fixed to zero, and the solver proceeds by replacing one basic variable by a non-basic one until the optimal solution is found. Variables are represented as indexed columns of A. The index of a non-basic column which enters the basis is determined according to the following expression [81]:

$$j \in \{1, \ldots, |N|\} \text{ s.t. } c_j - u^T A^j \text{ is maximal} \qquad (5.3)$$

[1] Simplex method is the most commonly used linear programming algorithm, see [32] for a detailed presentation.

where c_j is the objective coefficient for the new variable and u^T is the current dual solution of (5.2). The expression $c_j - u^T A^j$ is called *reduced cost*. At every iteration the column having the highest positive reduced cost is selected. If no such column exists the linear program is at an optimum and the simplex algorithm stops.

If the size of N is far too large, as is the case for the program (5.1), one should compute the index of the entering column according to (5.3) without examining all columns in N. This is done using the column generation technique in which (5.3) is treated as an optimization problem with the following objective function:

$$max\ (c_j - \sum_{i=1}^{m+1} u_i a_i^j),\ A^j = (a_i^j) \in \{0,1\}^{m+1} \tag{5.4}$$

where a_i^j are binary variables that represent linear coefficients of the entering column.

It is important to note that except for the way the entering column is obtained (i.e., generated vs selected) the simplex algorithm works along the same lines. Whether the column generation technique is successful or not is contingent upon the following criteria: i) there exists an efficient algorithm for the optimization problem (5.4), ii) an optimal solution of the program (5.2) can be found without the generation of an excessive number of columns (the number of generated columns characterizes *convergence* of the algorithm). In the next two subsections we present the PSAT algorithm and the optimized procedure which it uses to generate improving columns.

5.1.2 Column Generation-Based PSAT Algorithm

In order to explain the PSAT algorithm we first rewrite the linear system (5.1) as the following linear program:

$$\begin{aligned} \max\ & \sum_{W \in \mathcal{W}_\Phi} x_W \\ \text{s.t.}\ & \sum_{W \models C_i} x_W = p_i \times \sum_{W \in \mathcal{W}_\Phi} x_W,\ \text{for each } (C_i|\top)[p_i, p_i] \in \mathcal{P} \\ & \sum_{W \in \mathcal{W}_\Phi} x_W \leq 1 \text{ and all } x_W \geq 0 \end{aligned} \tag{5.5}$$

This program has the optimal objective value of 1 if and only if the system (5.1) is feasible. The advantage of using this program is that it is feasible even if the PTBox is not satisfiable which facilitates use of the column generation technique.[2] Algorithm

[2]In principle, it is possible to generate columns for an infeasible linear program but maintaining it feasible usually helps convergence.

3 presents the PSAT algorithm based on column generation.

Algorithm 3: PSAT algorithm based on column generation

Input: PTBox $PT = (\mathcal{T}, \mathcal{P})$
Output: Yes if PT is satisfiable, No otherwise

```
1  if T is not satisfiable then return No
2  LP ← InitializeRMP(T, P)
3  while true do
4  |   d ← optimize(LP)
5  |   u^T ← dual_solution(LP)
6  |   A^j ← GenerateImprovingColumn((T, P), u^T)
7  |   if A^j = null then break
8  |   else
9  |   |   Add A^j to LP as a new column
10 |   end
11 end
12 if d = 1 then return Yes else return No
```

The algorithm follows the basic column generation procedure outlined in the previous subsection. It first constructs so called *restricted master problem* (RMP) which is a subprogram of (5.5) with a restricted set of variables (line 2). These initial variables are created by generating a subset of the index set $\mathcal{W}_\Phi$ (see Section 2.3.1). Next the algorithm enters the main column generation loop (lines 3–10) during which it tries to generate an improving column (line 6). The column generation procedure *GenerateImprovingColumn*, which takes the PTBox and the current dual solution u^T, plays the central role and is explained in detail in the next subsection. If an improving column has been successfully generated, it is added to the linear program (line 9). The algorithm breaks out of the loop when no improving column can be generated. Finally, it checks the optimal value of the final RMP and returns Yes if it is equal to 1.

A number of implementation details have been omitted for the sake of presentation clarity. In particular, the algorithm forgets some previously generated columns which have not been in the basis for a large number of iterations in order to keep RMP tractable. This may potentially compromise the termination property (see the next subsection), so the algorithm implements special checks to detect the situation when previously forgotten columns are repeatedly regenerated. Also, we use stabilization techniques to improve convergence of the column generation process. Some details can be found in Section 5.1.5.

5.1.3 Possible World Generation

This section explains the procedure for generating improving columns (possible worlds) for Algorithm 3. It describes in detail what each component of a PSAT column represents and how to set up and use the optimization problem 5.4 for generation of possible worlds.

Consider a_i^j, the i-th coefficient of some column A^j for the PSAT program 5.5. The column corresponds to some possible world $W^j = \{C_i\}$, therefore $a_i^j = 1$ implies that C_i occurs positively in W^j while $a_i^j = 0$ implies that it occurs negatively (or equivalently, $W^j \models \neg C_i$). Thus it is possible to represent W^j as a *conjunctive* concept expression in $\mathcal{SROIQ}$ assuming a fixed linear ordering of concepts $\{C_i\}$ in Φ (see Section 2.3.1). More formally, we define the following function η which maps columns, i.e. binary vectors, to conjunctions of basic concepts from Φ:

$$\eta(A^j) = \bigsqcap X_i, \text{ where } X_i = \begin{cases} C_i, & \text{if } a_i^j = 1 \\ \neg C_i, & \text{if } a_i^j = 0 \end{cases} \tag{5.6}$$

X_i are literals that denote either a basic concept or its negation.

Soundness of Algorithm 3 strongly depends on whether every solution of the optimization problem (5.4), which is added as a column to the main linear program (5.5), corresponds to a concept expression that is satisfiable w.r.t. $\mathcal{T}$, i.e., is a *possible* world. If this condition is true then soundness trivially follows because one may simply enumerate the set of all solutions (since the set of possible worlds is finite), so (5.5) will be equivalent to the original linear system (5.1). *Completeness* requires that every possible world for the given PTBox corresponds to some solution of (5.4). Therefore, for ensuring both soundness and completeness it is crucial to construct a set of constraints $\mathcal{H}$ for the problem (5.4) such that its set of solutions is in one-to-one correspondence with the set of all possible worlds $\mathcal{W}_\Phi$.

In what follows we will call columns which correspond to satisfiable expressions *valid* and the others *invalid*. More formally, given a $\mathcal{SROIQ}$ TBox $\mathcal{T}$, a column A^j is valid if $\mathcal{T} \not\models \eta(A^j) \sqsubseteq \bot$ and is invalid otherwise.

Validity can easily be ensured in the propositional case where each C_i is a clause. One possibility is to employ the well known formulation of SAT as a mixed-integer linear program (MILP) [84]. For example, if $C_i = c_1 \vee \neg c_2 \vee c_3$ then (5.4) will have the constraint $a_i = x_{c1} + (1 - x_{c2}) + x_{c3}$ where all variables x_{ck} are binary. In that case soundness and completeness follow from the reduction of SAT to MILP. Previously developed propositional PSAT algorithms take full advantage of that (see Section 2.2.2).

In the case of an expressive language, such as $\mathcal{SROIQ}$, there appears to be no

easy way of determining the set of constraints $\mathcal{H}$. Furthermore, it is unclear whether such a set is polynomial in the size of $\mathcal{T}$. Informally, $\mathcal{H}$ must capture every entailment, such as $\mathcal{T} \models C_i \sqcap \cdots \sqcap C_j \sqsubseteq \bot$ in order to prevent generation of any column A^j such that $C_i \sqcap \cdots \sqcap C_j$ is a conjunctive subexpression of $\eta(A^j)$. All such entailments can be computed in a naive way by checking satisfiability of all conjunctions $C_i \sqcap \cdots \sqcap C_j$ over Φ but this is no better than trying to construct the full linear system (5.1).

Instead, Pronto implements a novel *hybrid*, *iterative* procedure to compute $\mathcal{H}$ which can be summarized as follows:

Algorithm 4: Possible world generation algorithm

Input: PTBox $PT = (\mathcal{T}, \mathcal{P})$, current dual solution u^T of (5.5)
Output: New column A^j or *null*

```
1  IP_ColGen ← initialize the integer program (5.4) using u^T and P
2  H ← ∅
3  while A^j ≠ null do
4  |   Solve IP_ColGen subject to H to optimality
5  |   A^j ← some optimal solution of IP_ColGen
6  |   if A^j ≠ null then
7  |   |   if satisfiable(η(A^j), T) then
8  |   |   |   return A^j
9  |   |   end
10 |   |   H ← H ∪ inequalities that exclude A^j
11 |   end
12 end
13 return null
```

The key steps are 7 and 10. On step 7 the algorithm invokes a $\mathcal{SROIQ}$ reasoner (in our case, Pellet [177]) to determine if the computed column corresponds to a *possible* world. This is critical for soundness. If yes, the column is valid and returned. If no, the current set of constraints $\mathcal{H}$ needs to be extended to exclude A^j from the set of solutions to (5.4). This step deserves a more detailed explanation which we present by first defining the notion of the minimal unsatisfiable core for an unsatisfiable conjunctive concept expression.

Definition 5.1 (Unsatisfiable Core)**.** *Given a TBox $\mathcal{T}$ and unsatisfiable (w.r.t. $\mathcal{T}$) concept expression $\bigsqcap X_i$ represented as a set of conjuncts $X = \{X_i\}$, a minimal unsatisfiable subexpression (MUS) is a subset $X' = \{X'_i\} \subseteq \{X_i\}$ such that $\bigsqcap X_i$ is unsatisfiable w.r.t. $\mathcal{T}$ and any $X'' = \{X''_i\} \subset \{X'_i\}$ is satisfiable w.r.t. $\mathcal{T}$. The **unsatisfiable core** (UC) of $\bigsqcap X_i$ is the set of all its MUSes.*

Intuitively, our notion of UC for conjunctive $\mathcal{SROIQ}$ concepts corresponds to the standard notion of unsatisfiable core for propositional formulas in conjunctive normal form [140].

Each MUS can be regarded as a one "laconic justification" of the unsatisfiability of the original concept expression [85] (here "laconic" means that it contains no superfluous conjuncts). The UC is the set of all laconic justifications of the unsatisfiability. Clearly, to exclude the current, invalid column from the set of solutions to (5.4), it is sufficient to add to (5.4) a constraint that rules out *any* of the MUSes.

Next, we show how to translate MUSes into linear inequalities. A MUS is a set of conjuncts $\{X'_i\}$ each of which corresponds to a binary variable (observe that η, as defined in (5.6), is a bijective function). By a slight abuse of notation we write $a_i = \eta^{-1}(X'_i)$ to denote the variable that corresponds to C_i. Then given a MUS $X' = \{X'_i\}_{i=1}^k$ we add the following linear constraint:

$$\sum_{i=1}^{k} a_i \leq k-1, \text{ where } a_i = \begin{cases} \eta^{-1}(X'_i), & X'_i = C_i \\ 1 - \eta^{-1}(X'_i), & X_i = \neg C_i \end{cases} \tag{5.7}$$

If a conjunctive concept contains $\bigsqcap X_i$ as a subexpression then all binary expressions b_i, i.e. either a_i or $1 - a_i$ depending on whether X_i is a positive or a negative literal, are equal to 1. Therefore, $\sum_{i=1}^k a_i = k$ where k is the size of $\{X_i\}$. Constraining $\sum_{i=1}^k b_i$ to be less or equal to $k-1$ is equivalent to requiring at least one b_i to be equal to 0. According to the definition of η this is equivalent to removing of at least one conjunct from X' which makes it satisfiable (due to minimality of X', see Definition 5.1). Therefore, each of the constraints (5.7) is sufficient to exclude all columns, which correspond to concept expressions containing X', from the set of solutions to (5.4). Observe that the constraints do not exclude any columns which do *not* include X' since it is necessary to ensure completeness.

On step 8 the algorithm computes the unsatisfiable core for a concept expression that corresponds to the current solution of (5.4). Then it transforms each of the MUSes into a linear inequality according to (5.7) and adds them to the binary program (5.4).

We call our PSAT algorithm, which is composed of algorithms 3 and 4, "hybrid" because it combines invocations of an LP solver (to optimize (5.5)), MILP and $\mathcal{SROIQ}$ solvers (to optimize (5.4) and check satisfiability of concept expressions respectively). It is *iterative* because during the possible world generation phase it iteratively tightens the set of solutions to (5.4) until either a valid column is found or provably no such column exists.

Finally, we give a short example demonstrating our iterative technique for computing valid columns.

Example 5.1. *Consider a PTBox where* $\mathcal{T} = \{A \sqsubseteq \exists R.C, B \sqsubseteq \exists R.\neg C, \geq 2R.\top \sqsubseteq D\}$ *and* $\mathcal{P}$ *contains some probabilistic constraints over the ordered set* $\Phi = \{A, B, D\}$. *Algorithm 4 starts out with an empty set of linear constraints for (5.4). The list of*

binary variables for (5.4) is (x_A, x_B, x_D)*. Assume that at some iteration the algorithm generates the following column:* $A^j = (1, 1, 0, 1)$ *(the last component of any column is always equal to* 1 *because of the normalization row in (5.5)). Then* $\eta(A^j) = A \sqcap B \sqcap \neg D$*.*

It is not hard to see that $\mathcal{T} \models \eta(A^j) \sqsubseteq \bot$*. The reason is that any instance* o *of* $A \sqcap B$ *must have two* R*-successors (domain elements which are connected to* $o^{\mathcal{I}}$ *by* $R^{\mathcal{I}}$*). Moreover, they are necessarily distinct because one is an instance of* C *and another is an instance of* $\neg C$*. Therefore,* o *is an instance of* $\geq 2R.\top$ *and consequently is an instance of* D*. This is a contradiction with* $\neg D$ *in* $\eta(A^j)$*.*

The unsatisfiable core of $\eta(A^j)$ *is* $\{A, B, \neg D\}$*. This MUS is converted into the following linear inequality* $x_D \geq x_A + x_B - 1$ *which is then added to the binary program (5.4). As a result, no invalid column containing this MUS will be computed on subsequent iterations.*

5.1.4 Algorithm Analysis

In this section we prove some basic properties of Algorithm 3 and investigate characteristics of PTBoxes which have a direct impact on performance of the PSAT algorithm presented in this section. For the subsequent analysis it will be convenient to distinguish two basic steps performed at each iteration of the algorithm: solving the master linear program (5.5) and generating the next column by invoking Algorithm 4. We refer to them as *Phase I* and *Phase II* respectively.

Theoretical Analysis We first prove that our algorithm is in fact a decision procedure for PSAT in P-$\mathcal{SROIQ}$ and then show that its worst case complexity is the same as for $\mathcal{SROIQ}$.

Theorem 5.1 (Soundness, Correctness, and Termination)**.** *Algorithm 3 which uses Algorithm 4 to generate columns is a sound, complete, and terminating procedure for* PSAT *in P-*$\mathcal{SROIQ}$*.*

Proof. Soundness. Every column added to the linear program (5.5) corresponds to a possible world, as ensured on step 5 (assuming that a sound $\mathcal{SROIQ}$ reasoner is used). Therefore, when the algorithm terminates, if the objective value of (5.5) is 1 then the final solution is a probability distribution over possible worlds which satisfies all conditional constraints, i.e. is a model.

Completeness. Assume that given a satisfiable PTBox $PT = (\mathcal{T}, \mathcal{P})$ the algorithm failed to prove its satisfiability. Consider the linear program of the form (5.5) whose optimal objective value is 1 and which was not constructed by the algorithm. There must be at least one basic column A^j, which corresponds to a variable having a positive value, but which was not generated by the algorithm 4. Therefore A^j is not a

solution of the column generation program (5.4) given its set of inequalities $\mathcal{H}$.[3] Consider any constraint $h \in \mathcal{H}$ such that A^j violates h. It is of the form $\sum_{i=1}^{k} b_i \leq k$ (see above) and, consequently, all k expressions b_i in A^j must be equal to 1. However, this implies that $\eta(A^j)$ contains an unsatisfiable conjunctive subexpression $X' = \{X_i\}_{i=1}^{k}$ (see (5.7)) which makes A^j invalid, thus we have a contradiction.

Termination. The PSAT algorithm terminates if i) the possible world algorithm terminates, and ii) each possible world is generated only once. The second condition is trivially satisfied because once a column has entered the basis, its reduced cost is zero and Algorithm 4 will not generate it again. To prove termination of Algorithm 4 observe that its maximal number of iterations corresponds to the number of MUSes of an unsatisfiable conjunctive expression. Each expression is finite, and so must be its set of unsatisfiable subexpressions. □

It is not hard to see that Algorithm 4 for P-$\mathcal{SROIQ}$ is an `N2ExpTime` procedure. Its running time is dominated by Phase II since solving the primal linear program can be done in polynomial time (in the size of the PTBox). Generation of each valid column takes no more than exponential (in the length of the probabilistic signature) number of SAT tests and the MILP optimization steps. Concept SAT in $\mathcal{SROIQ}$ is `N2ExpTime`-complete [106] while MILP is `NP`-complete in the size of the program (5.4). The latter is no greater than exponential in the size of the classical part of the PTBox. Thus the overall worst case complexity is `N2ExpTime`, i.e., the same as for optimal SAT algorithms for $\mathcal{SROIQ}$.

Practical Analysis Next we turn our attention to factors that are likely to influence the running behavior of the PSAT algorithm. Complexity of Phase I is determined solely by the size of the linear program (5.5). The number of inequalities is $m+1$ where m is the number of constraints in the PTBox.[4] In the worst case the number of variables is exponential in the size of probabilistic signature since it corresponds to the number of possible worlds. However, it is not necessary to always keep all the variables (columns) produced on the past iterations. It is common for column generation algorithms to "forget" variables which have not been a part of the basis over the last few iterations thus keeping the system small. Our algorithm follows this idea to dropping columns with the largest negative reduced cost every 50 iterations.

As such it can be concluded that the only important factors for the running time of Phase I are the PTBox's size and the size of the probabilistic signature. Possible world generation, or Phase II, is more complicated. Its running time depends on the following factors:

[3] If A^j was a solution of (5.4) then it would have a positive reduced cost and would eventually be chosen by simplex to enter the basis.

[4] It is $2 * m + 1$ when interval constraints are considered.

- The running time of optimizing the MILP program (5.4).
- The running time of deciding validity of each column candidate and computing the unsatisfiability core for conjunctive concept expressions in the target DL.
- The average number of re-optimizations of the MILP program required to produce a valid column.

We analyze how size and other characteristics of the input PTBox may influence the performance in practice. First, we consider the width of the MILP program (5.4). It is determined solely by the size of the probabilistic signature since each concept name in the signature corresponds to a distinct variable in the program.

The key factor determining height (the number of inequalities) of the MILP program and the complexity of computing the unsatisfiability core for conjunctive expressions is what we loosely call *TBox richness*. It is similar to various notions of axiomatic richness of DL TBoxes which appear in literature in different contexts, for example, in the context of modularity of ontologies [46]. What is common across these notions is that they reflect the number of non-trivial axioms entailed by the TBox. Unfortunately, to the best of our knowledge, neither of these notions has ever been made precise or widely applicable.

For our purposes it is handy to define richness in terms of the size and the structure of the space of possible worlds that satisfy the TBox. Since each axiom in the TBox makes some of the worlds impossible, one may think that axiomatic richness can be characterized simply by the number of possible worlds. The more worlds are ruled out by the structure of the TBox, the richer it is. For example, given the signature $\{A, B, C, D\}$ the TBox $\mathcal{T}_1 = \{A \sqsubseteq B, C \sqsubseteq D\}$ is richer than the TBox $\mathcal{T}_2 = \{B \sqsubseteq \neg C\}$ because $\mathcal{T}_2$ is satisfied by 12 worlds while $\mathcal{T}_1$ by 8 (out of 16). However, this method is too coarse. Consider the TBox $\mathcal{T}_3 = \{A \sqcap B \sqcap C \sqcap D \sqsubseteq \bot, A \sqcap B \sqcap \neg C \sqcap \neg D \sqsubseteq \bot, \neg A \sqcap B \sqcap C \sqcap \neg D \sqsubseteq \bot\}$. It rules out fewer worlds than either $\mathcal{T}_1$ or $\mathcal{T}_2$ but its structure is more complicated in the sense that more inequalities are required to faithfully capture it in the program (5.4).

To define TBox's richness we propose the following canonical way of describing the set of possible worlds for a TBox.

Definition 5.2 (Possible World Description)**.** *Given a TBox $\mathcal{T}$ its **possible world description** with respect to signature Φ is a set of concept subsumption axioms $\mathcal{C}_\Phi = \{C_i \sqsubseteq D_i\}_{i \in 1,\ldots,k}$, which satisfies the following conditions:*

i) *Each C_i and D_i are conjunctions of concepts from Φ and their negations.*

ii) *Each world W over Φ satisfies $\mathcal{C}_\Phi$ iff it satisfies $\mathcal{T}$.*

iii) *No axiom is a logical consequence of another, i.e., if $\alpha, \beta \in \mathcal{C}_\Phi$ then $\alpha \not\models \beta$ and vice versa.*

A possible world description exists for each TBox. One way of constructing it is to compute the set of all *impossible* worlds and create the axiom $\bigsqcap_i C_i \sqsubseteq \bot$ for each of them (while filtering out logical consequences to satisfy iii)). The size of the largest possible description is finite because the length of each expression is bounded by the size of the signature. With such definition at hand we can formulate the axiomatic richness of a TBox as follows:

Definition 5.3 (TBox Richness)**.** ***Axiomatic richness*** *of a TBox $\mathcal{T}$ with respect to a non-empty signature Φ, denoted as $K_\Phi(\mathcal{T})$, is the number of axioms in its maximal possible world description divided by $|\Phi|$.*

Formulated in this way TBox richness directly determines the maximal number of linear inequalities for the MILP program (5.4). It places an upper bound on the height of the program (5.4) but the column generation process may terminate before reaching it. According to our empirical experience, richer TBoxes always produce larger and, consequently, computationally harder column generation models. However we leave it to future work to assess whether significant portions of possible world descriptions are not yet captured in the MILP model when the process terminates.

In summary, we have identified signature size, PTBox size and TBox richness as the main factors which may influence the runtime performance of the PSAT algorithm. Chapter 6 presents the empirical evaluation results that characterize scalability and robustness of the algorithm on PTBoxes with variable parameters.

5.1.5 Main Optimizations

We next describe several optimization techniques which play key roles for our implementation of the possible world generation algorithm.

Classical Modularity

It is possible that concepts appearing in conditional constraints, i.e., the probabilistic signature, are only a small fraction of those appearing in the classical part of the KB. This can happen especially when a large OWL ontology, such as the NCI Thesaurus, is only slightly augmented with probabilistic knowledge. Then it seems intuitively unreasonable to work with the full classical part when solving PSAT/TLogEnt because many OWL axioms do not interact with probabilistic knowledge but only slow down concept satisfiability checks during the column generation process.

Fortunately, we can employ *ontology modularity* techniques in order to extract a fragment of the classical part which is guaranteed to contain all relevant knowledge, i.e., a module [69, 173, 46]. More formally, given a signature Σ a $\Sigma-$module $\mathcal{O}'$ in an ontology $\mathcal{O}$, written as $\mathcal{M}(\mathcal{O}, \Sigma)$, is a subset of $\mathcal{O}$ such that any axiom α over Σ is

entailed by $\mathcal{O}$ *iif* it is entailed by $\mathcal{O}'$ [69]. In the context of PSAT this means that any concept expression 5.6 that needs to be checked for satisfiability on step 10 of Algorithm 4 is unsatisfiable w.r.t. the whole classical part $\mathcal{T}$ *iff* it is unsatisfiable w.r.t. $\mathcal{M}(\mathcal{T}, \Phi)$, where Φ is the probabilistic signature.

Extracting modules enables us to substantially cut down the size of the classical part even though modules are not guaranteed to be minimal.[5] In fact, this optimization is not specific for our hybrid algorithm and even column generation. In particular, it can be used in propositional PSAT solvers as well. Note that modules are only guaranteed to contain all relevant knowledge w.r.t. a *fixed* signature so if the signature changes, for example, when a new conditional constraint is added or some new concept is used in an entailment query, the module needs to be recomputed.

Exploiting Concept Hierarchy

The first optimization stems from a natural observation that many inequalities for the binary program (5.4) can added simply by examining the structure of a TBox. Virtually all modern DL reasoners can efficiently construct the so called *classification hierarchy* by finding all subsumptions between concept names that are logically entailed by the TBox. Such hierarchy can be used to construct an initial set of inequalities $\mathcal{H}_0$.

Consider the following TBox $\mathcal{T} = \{A \sqcup B \sqsubseteq C\}$. The classified version of $\mathcal{T}$ should include subsumptions $A \sqsubseteq C$ and $B \sqsubseteq C$. They can be directly translated to inequalities $x_A \leq x_C$ and $x_B \leq x_C$ to preclude computing an invalid column containing either $A \sqcap \neg C$ or $B \sqcap \neg C$ as subexpressions and converting these subexpressions into inequalities.

This idea helps to reduce the number of concept satisfiability tests. The effectiveness of this technique depends on the axiomatic richness of the TBox. For axiomatically weak TBoxes, where almost all subsumptions can be discovered by traversing the concept hierarchy, most of the set $\mathcal{H}$ is computed up front. More complex TBoxes may have non-trivial entailments involving concept expressions on both sides of subsumptions which can only be discovered when checking validity of some column candidate. One such examples is the subsumption $A \sqcap B \sqsubseteq D$ from Example 5.1.

A drawback of this optimization is that it might be *too eager* and generate more linear inequalities that can be fit in memory. One example of how this can happen is exploiting TBoxes which succinctly encode a quadratic number of disjointness axioms by using the `DisjointClasses` construct in OWL 2.[6] At the same time, as explained

[5] Extracting strictly minimal models is undecidable for $\mathcal{SROIQ}$ so we rely on approximate solutions. See [69] for details.

[6] The axiom `DisjointClasses(`C_1`,...,`CE_n`)` asserts that the concepts are *pairwise* disjoint. It is equivalent to a quadratic number of binary disjointness axioms. Such axioms can be added by ontology editors transparently for the user. For more details refer to `http://www.w3.org/TR/owl2-syntax/#Disjoint_Classes`

in the next subsection, there is a chance that enough valid columns can be generated *before* all the relationships entailed by the TBox are discovered and captured in linear constraints. Therefore a simple solution to this problem is to impose a limit on the total number of inequalities created up front. The value of the limit can be adapted depending on the amount of available memory and capabilities of the MILP solver. A more informative approach could be possible and is left for future investigation.

Propositional Absorption

A large portion of knowledge in many real ontologies is propositional. Thus it is natural to convert them into linear inequalities to avoid computing some invalid columns which violate propositional TBox axioms. In the extreme case all propositional knowledge can be absorbed into the program (5.4). However, the algorithm tries to find a trade-off between eager absorption (which can exhaust memory) and lazy generation of inequalities (which requires extra concept satisfiability checks). The balance depends on available memory and the number of propositional axioms.

This optimization can add extra variables to the column generation model (5.4). Normally, its variables correspond to concepts in the probabilistic signature Φ. Consider an example in which $\{C_1, C_2, C_3\} \subseteq \Phi$. If C_1 is itself defined as a Boolean combination over other concepts, e.g., $C_1 \equiv A \sqcup B$, it makes sense to add x_A and x_B to (5.4) (even if A and B do not appear in any conditional constraints). The reason is that other concepts from Φ could appear in propositional expressions over $\{A, B, \dots\}$, for instance, $C_2 \equiv \neg A \sqcup B, B \sqsubseteq C_3$. In this case, adding extra variables and translating these axioms into linear inequalities will *automatically* enforce $C_1 \sqcap C_2 \sqsubseteq C_3$ for all future column candidates.

Optimistic Inequality Generation

One issue with a naive implementation of Algorithm 4 is that computing unsatisfiability cores may appear impractical for certain concept expressions and TBoxes. This may especially happen for long expressions which contain MUSes with little or no overlap. It is known from the model diagnosis theory [169] that finding all minimal unsatisfiable sets may require a number of SAT tests that is exponential in the total size of all sets.

To address this issue the algorithm imposes a time limit on the procedure that computes the UC. If at least one MUS has been found but finding others exceeds the timeout the procedure is interrupted. The found MUSes are then converted to linear inequalities and the algorithm moves on as if the full UC was computed.

This optimization does not cause a loss of either soundness or completeness. Completeness is trivially preserved because not adding some inequalities to the program

(5.4) can only expand its set of solutions, so no possible world could be missed. Soundness is preserved because each computed column is still valid (SAT tests are never interrupted). The only possible negative impact of missing some MUSes is that they can appear in some future column candidates, so the algorithm might go through additional iterations. However, they do not *have to* appear because the optimal basis for the main program (5.5) can often be found before considering column candidates containing those MUSes. Intuitively, the algorithm behaves *optimistically* by hoping that additional iterations will not be required.

Unsurprisingly, timeouts typically occur when dealing with complex TBoxes with many non-trivial entailments. At the same time our experience shows that those TBoxes tend to improve convergence of column generation because their set of possible worlds is smaller. Therefore there is a higher chance that the column generation process will stop before all MUSes of some unsatisfiable concept expression are discovered.

Multiple Column Generation and Stabilization

We use several techniques aimed at improving convergence of the column generation process. The most important are generating several optimal solutions of (5.4), i.e. column candidates, and introducing additional variables for the main linear program (5.5) to stabilize dual space and reduce degeneracy.

It is known that adding multiple columns into basis at every simplex iteration can improve convergence of column generation. For instance, Hansen and Perron add up to 50 columns per iteration [81]. This is made possible by their heuristic method of generating columns (the variable neighborhood heuristics) which allows them to quickly generate many sub-optimal columns having positive reduced cost. In our case, when columns are generated by a MILP solver, there are the following two possibilities:

- Some MILP solvers, in particular, CPLEX, support *solution pools* which store multiple optimal or sub-optimal solutions for an MILP problem instance. Such solvers can either return the set of sub-optimal solutions which they recorded during the optimization process or continue the branch-and-cut search after the optimum has been found until the required number of solutions has been found.
- If the solver does not support solution pools one may still force it generate multiple solutions. This can be done by "cutting off" the optimal solution and re-optimizing or by hooking inside the solver to continue the search after the optimum.

Pronto is highly modular and can be used with different LP/MILP solvers. Using a solution pool is the first choice strategy if it is available. If it is not available, as is the case with GLPK, then the second option is used. However, since it has implications for

performance, fewer columns are generated (usually up to 10). More details on methods for enumerating solutions for a MILP instance can be found in [41].

We use two different stabilization schemes in Algorithm 3. First, similarly to [81] we use the iterative technique proposed by du Merle et al. [51] to stabilize the values of the dual variables and reduce degeneracy.[7] This involves adding extra variables to the main linear program so it looks as follows:

$$\max \sum_{W \in \mathcal{W}_\Phi} x_W + \delta^+ y_0^+ - \delta^- y_0^- + \delta^+ y^+ - \delta^- y^- \tag{5.8}$$

$$\text{s.t.} \sum_{W \models C_i} x_W = p_i \times \sum_{W \in \mathcal{W}_\Phi} x_W + y^+ - y^-$$

$$\sum_{W \in \mathcal{W}_\Phi} x_W + y_0^+ - y_0^- \leq 1 \tag{5.9}$$

$$y_0^+, y^+ \leq \epsilon^+, y_0^-, y^- \leq \epsilon^-$$

$$x_W, y^+, y^-, y_0^+, y_0^- \geq 0$$

where y^+ and y^- are the column vectors $(y_1^+, \ldots, y_m^+)^T$, $(y_1^-, \ldots, y_m^-)^T$ respectively. They are bounded by positive values of ϵ^+,ϵ^- and have objective coefficients δ^+,δ^-. The parameters δ^+,δ^- correspond to bounds on dual variables while ϵ^+,ϵ^- correspond to penalties on violation of those bounds. ϵ^+,ϵ^- are progressively reduced to 0 every time (5.8) is solved to optimality. Once they are zero, the optimal value of the augmented program is the same as of the original PSAT program (5.5). More details about stabilization of column generation can be found in [51, 81]. Our stabilization is a little different from the one implemented by Hansen and Perron [81]. In particular, we use scalar parameters $\epsilon^+, \epsilon^-, \delta^+, \delta^-$ instead of vectors and also vary the number of extra variables.

Second, we use the non-iterative technique in which the main linear program looks slightly simpler:

[7] Degeneracy is the situation when some basic variables have zero values. It is known to be a problem for simplex since it can lead to multiple column swaps without an improvement of the objective value.

$$\max \sum_{W \in \mathcal{W}_\Phi} x_W - \delta_0 y_0 - \delta^+ y^+ - \delta^- y^- \tag{5.10}$$

$$\text{s.t.} \sum_{W \models C_i} x_W = p_i \times \sum_{W \in \mathcal{W}_\Phi} x_W + y^+ - y^-$$

$$\sum_{W \in \mathcal{W}_\Phi} x_W + y_0^+ - y_0^- \leq 1 \tag{5.11}$$

$$x_W, y^+, y^-, y_0, \delta^+, \delta^- \geq 0$$

For this stabilization it is not required to iteratively reduce bounds on new variables to zero because the program can *only* reach the objective value of 1 when all extra variables are equal to 0. Conversely, if when the column generation process terminates the objective value is less than 1, tightening bounds on extra variables cannot improve it. Therefore this stabilization program can be used directly for solving PSAT. Of course, the extra variables have to be removed when computing TLogEnt.

We leave it for future research to investigate under which conditions one stabilization scheme is superior to another. In most of our experiments reported in Chapter 6 the non-iterative technique is used. However, there are cases when it is not effective although they are difficult to predict. Fortunately, it is possible to detect stalling of the column generation process (when a little progress is being made and the early clash heuristics does not prove unsatisfiability) and switch to the iterative method.

Early Unsatisfiability Detection

Automated reasoners often implement heuristic approaches to quickly detect obvious reasons for unsatisfiability of logical formulae. One good example of such techniques is the early clash detection methods employed by virtually all mature DL reasoners. Probabilistic reasoners are no exception, for instance, propositional PSAT solvers sometimes use incomplete rule-based reasoning methods which prove to be tremendously effective at proving unsatisfiability [81].

Since rule sets have only been formulated in the propositional case we take another approach. It is based on the observation that if the PSAT program has an optimal objective value of 1 (i.e. the KB is satisfiable) then the optimal dual solution is of the form $(0, \ldots, 0, 1)$. The dual values show the rate at which the objective value of the primal program will improve in response to a small relaxation of row bounds. If the optimal value of the program (5.5) is 1 then the only row whose relaxation can improve the objective is the last one, i.e. $\sum_{W \in \mathcal{W}_\Phi} x_W \leq 1$. Its coefficients are exactly the same as the objective coefficients, thus its dual value is 1 while the other dual values are zeros. On the other hand, if the optimal objective value is below 1 then the indexes of

non-zero dual variables indicate conflicting inequalities and, consequently, conflicting conditional constraints.

Our algorithm maintains a history of dual solutions over a fixed number of column generation iterations. The history helps to spot the situation when the *same* dual variables repeatedly take on non-zero values while the primal objective is improving very slowly. More precisely, if over the last 10 iterations:

- The objective function improved by less than 1% and,
- Fewer than 10% of dual values had non-zero values at *all* iterations

then it is a good indication that the set of conditional constraints, which correspond to those dual variables, is not satisfiable. In that case, satisfiability of such constraints is tested separately. If they are indeed unsatisfiable, the whole process stops and unsatisfiability is reported, otherwise the main column generation cycle continues.

The technique preserves *soundness* because of monotonicity: if some subset of a KB is unsatisfiable, then the whole KB is unsatisfiable. It also preserves *completeness* since for each satisfiable KB the column generation process will necessarily continue until satisfiability is proved (all extra tests, if any, must be negative).

The heuristic is very effective for unsatisfiable KBs. In our experiments more than 90% of unsatisfiable KBs have been proved unsatisfiable by this method. It is effective mostly because conflicting sets of constraints tend to be small so it can easily be spotted if a small number of duals progress towards non-zero values. However, if the KB is satisfiable then the heuristic can slow the column generation down by introducing extra tests but this has so far happened in less than 15% of the cases.

5.1.6 Comparison with Propositional PSAT

This section succinctly summarizes the key differences between our PSAT algorithm and the previously developed *global* algorithms for propositional PSAT (see Section 2.4.1 for an overview). Those algorithms are similar in spirit since they are also based on the column generation technique but are different in scope, design, and implementation. The major differences manifest themselves in the way the algorithm deals with classical part of a probabilistic knowledge base. We consider two cases: when classical knowledge is propositional and when it is not.

To the best of our knowledge, our work is the first to describe a column generation based algorithm and its evaluation for *non-propositional* PSAT. The main difference between propositional and non-propositional PSAT problems in the context of column generation is that propositionality of the KB allows encoding of all its structure in a polynomial number of linear inequalities over a polynomial number of binary variables. This follows directly from the well known reduction of propositional SAT to integer

programming [84, 83]. Therefore, the column generation problem (5.4) is much easier to handle, either as a standard MILP instance [84, 45] or as a non-linear pseudo-boolean program [99, 81, 159]. The full structure of a $\mathcal{SROIQ}$ TBox, on the other hand, cannot be captured as a system of linear inequalities.[8]

The key property of our algorithm that enables it to deal with non-propositional KBs is its "hybridness". It interacts with a classical SAT solver for the target logic in order to ensure that all generated columns (possible worlds) are indeed possible, i.e., do not violate logical structure of the KB, which can be quite rich. This has several important advantages. First, it allows us to handle essentially *any* target logic for which a SAT solver is available, for example, we can use specialized reasoners for any profile of OWL 2. Second, this makes our algorithm more scalable with respect to the amount of classical knowledge. For example, modern DL reasoners can efficiently solve the concept satisfiability problem even for very large TBoxes containing thousands of axioms (a characteristic example is the NCI Thesaurus). The reasoners implement a range of optimizations and heavily exploit the logical structure of the KB, which is often lost if translated to linear inequalities (even when the translation is possible).

The behavior of our algorithm becomes closer to that of the previous methods when the classical knowledge is largely propositional. In that case, it can *absorb* the propositional part of the KB by converting it into a set of linear inequalities for the column generation problem (5.4), i.e., similarly to [84]. This helps to reduce the number of future calls to the classical SAT solver.

In runtime the algorithm maintains the balance between the *pure absorption* strategy, when all classical propositional formulas are converted into linear inequalities, and the *pure interaction* strategy, when the the column generation problem (5.4) is initially unconstrained and all inequalities are added only when some produced column candidate appears to correspond to an unsatisfiable concept expression (see line 10 of Algorithm 4). The former minimizes the number of calls to the classical SAT solver at the cost of large MILP programs. The latter minimizes the size of (5.4) by heavily interacting with the SAT solver in order to capture only relevant parts of the logical structure in linear inequalities. The balance is partly mandated by the kind of classical knowledge, in particular, its richness and size, as well as the amount of available memory. Evaluation results presented in Section 6.5 illustrate the influence of richness and size on performance of Algorithm 4.

Our PSAT algorithm is a substantial improvement of the earlier described algorithm [111] which is based on a similar idea but generates columns by solving a generic constraint optimization problem. The MILP formulation, presented in Section 5.1.3,

[8] By "capture" we mean that there is no polynomial reduction of SAT in $\mathcal{SROIQ}$ to the solvability problem for a system of linear inequalities over integer variables or any other standard mathematical programming model.

appears to be much more tractable and amenable to optimizations which resulted in the scalability improvements of the order of magnitude.

5.2 Analysis of Probabilistic Knowledge Bases

The PSAT algorithm plays a fundamental role in P-$\mathcal{SROIQ}$ reasoning but it is insufficient for practical implementations of other reasoning procedures, in particular, consistency and tight lexicographic entailment. This is somewhat analogous to classical DLs: a scalable concept satisfiability (CSAT) algorithm is tremendously important but none of the state-of-the-art reasoner would compute the concept hierarchy by a straightforward reduction to CSAT (i.e., by solving a quadratic number of CSAT tests for every pair of concepts). Instead they implement specific algorithms for solving other reasoning tasks, such as concept classification, realization of individuals, etc.

Non-monotonicity of P-$\mathcal{SROIQ}$ makes it even harder to expect that straightforward reductions of PTCon and TLexEnt to PSAT and TLogEnt could be practical. This is especially true for the original TLexEnt algorithm which requires an exponential number of PSAT tests. The problem is that PSAT, being a decision problem, does not provide much insight into the knowledge base, for example, why a certain KB is unsatisfiable. This makes PSAT too coarse an instrument for constructing z-partitions and computing lex-minimal subsets of constraints, and leads to a unnecessarily large number of PSAT tests to be performed during non-monotonic reasoning.

This section describes a solution to this problem based on a more fine-grained analysis of the knowledge base, in particular, for pinpointing conflicts between conditional constraints. As will be shown below such analysis has important applications for constructing and maintaining z-partitions, computing lex-minimal subsets, explaining unsatisfiability and so forth.

5.2.1 Finding Minimal Unsatisfiable Subsets

First, we present the approach for finding conflicts in the probabilistic part of a PTBox. The notion of a conflict, which has been used rather loosely up to now, can be formalized as follows:

Definition 5.4 (Minimal conflict)**.** *A minimal probabilistically unsatisfiable subset of a PTBox $(\mathcal{T},\mathcal{P})$ (where $\mathcal{T}$ is a consistent $\mathcal{SROIQ}$ TBox), or a* ***minimal conflict*** *in $\mathcal{P}$, is a a subset $\mathcal{P}' \subseteq \mathcal{P}$ such that i) $(\mathcal{T},\mathcal{P}')$ is unsatisfiable and ii) $(\mathcal{T},\mathcal{P}'')$ is satisfiable for any $\mathcal{P}'' \subset \mathcal{P}'$. The set of all minimal conflicts of PT is denoted as $MC(PT)$.*

In what follows the term "conflict" will mean "minimal conflict" (unless explicitly stated otherwise). The following problem, which we call the *Diagnosis problem*, plays

a fundamental role in our implementations of PTCon and TLexEnt reasoning procedures.

Definition 5.5 (Diagnosis problem)**.** *Given an unsatisfiable PTBox* $(\mathcal{T}, \mathcal{P})$ *(where* $\mathcal{T}$ *is a consistent* $\mathcal{SROIQ}$ *TBox), compute the set of all minimal conflicts in* $\mathcal{P}$*.*

The Diagnosis problem is closely related to the model-based diagnosis theory as originally developed by Reiter [169]. It is basically a probabilistic analogue of the problem of finding all justifications of unsatisfiability in DL knowledge bases which recently gained an extensive research attention [103, 124, 85, 86]. Historically there have been two main ways of dealing with this problem:

Black-box reasoning assumes that only a satisfiability solver for a target logic, e.g. P-$\mathcal{SROIQ}$ or $\mathcal{SROIQ}$, is available. The conflicts are found by first finding *some* conflict in the KB (this can be done by removing the statements until the KB becomes satisfiable), and then systematically exploring the hitting set tree of the current set of conflicts. The process terminates after all possible repairs for the current set of conflicts have been tried and none of them led to discovery of a new conflict. Details can be found in [169, 85].

Glass-box reasoning assumes that the SAT solver for a target logic provides some additional information about what sets of statements may have caused unsatisfiability of the KB. Such information can be used to speed up the conflict discovery process. For example, tableaux reasoners can report the information about clashes occurred inside the tableau (this capability is sometimes called *tableau tracing* and is supported by some $\mathcal{SROIQ}$ reasoners, e.g. Pellet [177]).

The black-box approach is easier to implement but generally tends to require a higher number of satisfiability tests. Therefore we have developed a mixed approach which follows the basic black-box strategy but with two important differences. First, it makes use of additional information provided by simplex solvers to efficiently identify conflicting statements in an infeasible system. Second, it maintains a single set of generated columns across all PSAT instances solved to find all conflicts.

Identifying Some Minimal Conflict

A straightforward way to find some conflict in a PTBox $PT = (\mathcal{P}, \mathcal{T})$ is to remove statements from $\mathcal{P}$ until PT becomes satisfiable. In the worst case this requires $|\mathcal{P}|$ PSAT checks. More efficient variations are possible, for example, one may implement a binary search for conflicts in $\mathcal{P}$ (such approaches are called *contraction strategies* in [86]) thus reducing the number of PSAT tests to $O(log|\mathcal{P}|)$ in practical cases. Such approaches, while different, are inherently black-box, i.e. they do not use any information from inside the PSAT solvers to determine *why* the PTBox is unsatisfiable.

However, if the PSAT solver uses the simplex algorithm to optimize the linear program (5.5), then it can provide some useful information which points to inequalities which prevent it from reaching the optimal objective value of 1. Virtually all modern simplex solvers support *sensitivity analysis* to provide insight into how the objective value of the program will change in response to small changes in linear inequalities. This is done by returning the vector of dual variables which values correspond to the inequalities of the primal program. The key fact is that if a dual variable y_i is zero then no small change in the right hand-side of the inequality $A_i x = p_i$ of (5.5) will affect the objective value. In other words, this inequality is not among those that prevent the program to improve its objective value. Assuming a fixed ordering of constraints in $\mathcal{P}$ (see Section 5.1), this means that the constraint $(C_i|\top)[p_i, p_i]$ is *not* necessarily in some conflict for the current PTBox.

Sensitivity analysis allows us to restrict our attention to those constraints in $\mathcal{P}$ which correspond to the inequalities having non-zero dual values. This greatly improves the effectiveness of finding some conflict. The only difficulty here is that the program (2.3) is not represented in its full form and neither is its dual. However, the following theorem shows that we may still use the dual solution obtained after the last iteration of the column generation algorithm.[9]

Theorem 5.2. *Let max* $1x$ *s.t.* $Ax \leq b$ *be a linear program whose optimal value is strictly less than* 1*. Assume that when the column generation process stops* $A = (B, N)$*, where* B *is the final RMP and* N *is the set of other columns (not produced by the column generator). Let* $u*$ *be the final dual solution of the program with* I *being the index set of non-zero components of* $u*$*. Then the optimal value of max* $1x$ *s.t.* $A_I x \leq b_I$*, where* A_I *(resp.* b_I*) is a sub-matrix of* A *(resp. sub-vector of* b*) having rows (resp. components) with indexes from* I*, is also strictly less than* 1*.*

The proof can be found in Appendix A.

Algorithm 5 for finding *some* minimal conflict in a PTBox is based on Theorem 5.2. It takes the PTBox and an already constructed linear program (5.5) because it is invoked repeatedly (see the next subsection) so reusing the same program improves performance. The algorithm goes through two main phases. The first phase (line 1) is essentially solving PSAT for the PTBox but starting with the initial linear program. If after the column generation process has been completed the optimal value of the linear program is still less than 1, the algorithm proceeds to the second phase (called "minimization"). The algorithm then extracts the sub-program LP' whose optimal objective value is less than 1 (lines 3–5). However, Theorem 5.2 does not claim that LP' is *irreducible*. In other words while $(\mathcal{T}, \mathcal{P}')$ is unsatisfiable, where $\mathcal{P}'$ is a subset of

[9] This idea came from a personal communication with Dr. Paul Rubin from the Graduate School of Management, Michigan State University.

Algorithm 5: Finding some conflict in PT

Input: Initial linear program LP, PTBox $(\mathcal{T}, \mathcal{P})$
Output: Minimal conflict of PT, or $\emptyset$ if the optimal objective value of LP is 1

```
   /* PSAT phase                                                    */
1  LP ← GenerateColumns(LP, (T, P))
2  if Objective value of LP is less than 1 then
3  |  duals ← optimal dual solution of LP
4  |  IS ← indexes of non-zero components of duals
   |  /* Minimization phase                                        */
5  |  LP' ← LP|_IS
6  |  for L ∈ inequalities(LP') do
7  |  |  LP' ← LP' \ L
8  |  |  if Objective value of LP' is 1 then LP' ← LP' ∪ L
9  |  end
10 |  return Subset of P that corresponds to the inequalities in LP'
11 end
12 else return ∅
```

$\mathcal{P}$ that corresponds to the inequalities in LP', it may be a strict superset of a minimal conflict. Thus the algorithm performs an extra trial-and-error relaxation step to remove superfluous inequalities from LP' and then returns the subset of $\mathcal{P}$ that corresponds to the remaining inequalities. With a slight abuse of notation, lines 7 and 8 remove an inequality from the program and then put it back in if it is essential. According to our empirical experience this minimization step is typically very quick as the number of superfluous inequalities is low.

Finding All Minimal Conflicts

Pinpointing a single minimal conflict of an unsatisfiable PTBox is a necessary but not sufficient tool to finding all such conflicts, i.e., solving the Diagnosis problem. The principled way of doing so, as developed in the generic theory of model-based diagnosis [169], is based on a systematic investigation of all ways to *repair* the current set of conflicts in order to discover new conflicts. This idea is widely used, for example, for computing explanations and repairing undesirable entailments in DL ontologies [104, 103, 85, 86]. Definition 5.6 makes the notion of repair precise before we proceed to explaining the algorithm.

Definition 5.6 (Repair)**.** *Given a set of conflicts $\mathcal{C}$ a **repair** R is a set which contains one constraint from each element of $\mathcal{C}$. A **minimal** repair R is a repair such that none of its proper subsets if a repair.*

In other words, a minimal repair is simply a minimal *hitting set* of $\mathcal{C}$. It is called *repair* because removing R from the PTBox restores its satisfiability provided that the

set of conflicts $\mathcal{C}$ is complete (see the next subsection). Example 5.2 illustrates the definition:

Example 5.2 (Repair and Minimal Repair)**.** *Let* $(\mathcal{T}, \mathcal{P})$ *be the following PTBox:* $\mathcal{T} = (\{A \sqsubseteq B, G \sqcap H \sqsubseteq \bot\}$, $\mathcal{P} = \{(F|B)[0.8, 0.9], (F|A)[0, 0.1], (E|B)[0.8, 0.9], (E|A)[0, 0.1], (A|\top)[1, 1], (G|H)[0.5, 0.7], (H|\top)[1, 1]\})$.

It contains three conflicts: $C_1 = \{F|B)[0.8, 0.9], (F|A)[0, 0.1], (A|\top)[1, 1]\}$, $C_2 = \{E|B)[0.8, 0.9], (E|A)[0, 0.1], (A|\top)[1, 1]\}$, *and* $C_3 = \{(G|H)[0.5, 0.7], (H|\top)[1, 1]\}$. *The conflicts* C_1 *and* C_2 *overlap at* $(A|\top)[1, 1]$ *but are disjoint with* C_3*, so minimal repairs (hitting sets) are those which contain* $(A|\top)[1, 1]$ *and one of the two constraints in* C_3.

Algorithm 6 applies this approach to the Diagnosis problem for P-$\mathcal{SROIQ}$ ontologies. It first constructs the initial linear program (5.5) for the given PTBox and then starts the main find-and-repair loop. It uses Algorithm 5 as a sub-procedure to extract a minimal conflict using the current linear program (line 6). Every time a new conflict is found it is added to the set $\mathcal{C}$ (line 9). When this happens the algorithm updates the set of repairs (or hitting sets) at line 10 so that each next repair also accounts for the newly added conflict. The algorithm continuously loops through the list of repairs applying each of them at line 5 and then restoring the unsatisfiability at line 7. The algorithm terminates when all repairs have been tried and none led to discovery of a new conflict (which would have triggered an update the list of repairs).

Algorithm 6 implements two important optimizations which are not shown for the sake of clarity. First, it attempts to reuse information obtained from checking previous repairs when applying each next repair. In particular, if repair R was applied and no new conflict was found (i.e. $(\mathcal{T}, \mathcal{P} \setminus R)$ is satisfiable) then all subsequent repairs that are supersets of R can be safely skipped. Second, it computes the set of repairs *incrementally*. The minimal hitting set problem is known to be `NP`-complete (it is equivalent to the vertex cover problem) so it is important to reuse previously computed hitting sets. We implemented two sub-algorithms for that tasks: one is based on binary hitting set trees [130] and the other is based on binary integer programming [56]. Both algorithms have incremental behavior: the trees can be updated incrementally while the modern IP solvers can "kick start" from a set of solutions and re-optimize the model once it has been changed. Pronto's current implementation uses the second approach.

In the worst case the algorithm makes $O(|\mathcal{HS}|)$ attempts to find a single conflict in a PTBox. The number of all possible repairs of a set of conflicts $\mathcal{C}$ is exponential in $|\mathcal{C}|$. Each repair leads to a single invocation of Algorithm 5 whose complexity, as mentioned earlier, is equivalent to the complexity of PSAT (modulo the minimization step). Since PSAT in P-$\mathcal{SROIQ}$ is `N2ExpTime`-complete so is the Diagnosis problem (the algorithm establishes the upper complexity bound while PSAT establishes the lower bound).

Algorithm 6: Diagnosis algorithm

Input: PTBox $PT = (\mathcal{T}, \mathcal{P})$
Output: The set of all minimal conflicts of PT

```
1  C ← ∅, HS ← ∅
2  LP ← initial linear program (5.5) for (T, P)
3  repeat
4      if HS ≠ ∅ then R ← remove some element from HS else R ← ∅
5      Free inequalities in LP that correspond to constraints in R
6      P' ← FindSomeConflict(LP, (T, P))
7      Enforce inequalities in LP that correspond to constraints in R
8      if P' ≠ ∅ then
9          C ← C ∪ P'
10         HS ← AllMinimalHittingSets(C)
11     end
12 until HS = ∅ ;
13 return C
```

Not surprisingly the number of repairs is the key factor for performance of Algorithm 6. This is verified empirically in Section 7.2. This number is higher for those PTBoxes whose conflicts do not overlap to a high extent, i.e., there are many relatively independent reasons for unsatisfiability. Given the potentially high number of PSAT tests it is important to re-use intermediate computation results across them. We do so by re-using columns generated for one PSAT for all subsequent PSATs. This can be done because all PTBoxes differ only in their probabilistic parts but share the same TBox, so columns which are valid for one linear program (2.3) will be valid for all others. This greatly speeds up the process of finding all conflicts.

Other implementations of the conflict finding procedure are possible and may be more efficient. In particular, one may attempt to employ known algorithms for enumerating all irreducible infeasible subsystems (IIS) of an infeasible linear system [66, 161]. They are generally based on the fact that for every infeasible system LS_{inf} there exists a linear system LS_{IIS} such that the set of all optimal solutions of LS_{IIS} is in one-to-one correspondence with the set of IISes for L_{inf} [66]. The main difficulty with this approach is that linear programs for PSAT are never fully represented. While the optimal RMP has the same objective value as the full program, its set of IISes[10] does *not* have to be the same as that of the full program. This is not very surprising, for example, a similar situation occurs when computing justifications of entailments from modules extracted from an ontology [46]. While all reasonable kinds of modules preserve entailments, not all of them guarantee to contain all justifications of entailments.[11] This

[10] IIS of the linear program 5.5 denotes an irreducible subprogram whose optimal objective value is strictly less than 1.

[11] Those which do have that property are called *depleting* modules [173].

difficulty can be overcome by generating extra rows for LS_{IIS} but, to the best of our knowledge, this option has not yet been investigated.

We conclude with the following theorem which states that the algorithm always terminates, computes all minimal conflicts and nothing but conflicts. The algorithm follows the classical diagnosis idea so the results follow quickly from correctness and termination of Algorithm 5

Theorem 5.3. *Algorithm 6 is a correct, complete, and terminating algorithm for solving the Diagnosis problem.*

Proof. Termination As mentioned above the algorithm invokes Algorithm 5 $O(|\mathcal{HS}|)$ times, which is a finite number. Algorithm 5 terminates (because the column generation process has to terminate) therefore so does Algorithm 6.

Correctness Correctness follows from correctness of Algorithm 5 which, in turn, is a consequence of Theorem 5.2.

Completeness Assume the algorithm is *not* complete, i.e. there is a conflict $\mathcal{P}'$ in $(\mathcal{T}, \mathcal{P})$ which is not an element of $\mathcal{C}$ (the set of conflicts computed by the algorithm). Let $\mathcal{HS}$ be the set of minimal hitting sets over $\mathcal{C}$ i.e., the *final* set of repairs checked by the algorithm before termination. There should be $h \in \mathcal{HS}$ s.t. $h \cap \mathcal{P}' = \emptyset$, otherwise either h or $\mathcal{P}'$ would not be minimal. If $\mathcal{P}'$ is a conflict then $(\mathcal{T}, \mathcal{P} \setminus h)$ must be unsatisfiable. But it is satisfiable because Algorithm 5, which is correct, did not find a conflict in it. Contradiction. □

5.2.2 Finding Maximal Satisfiable Subsets

The first immediate application of the diagnosis algorithm is computing all maximal satisfiable subsets of an unsatisfiable PTBox. This is essential, in particular, for computing lexicographically minimal models especially for PABox entailments when probabilistic facts about the individual contradict some PTBox constraints (see Section 5.4).

The following definition and the theorem formalize a straightforward connection between the set of minimal unsatisfiable subsets and the set of maximal satisfiable subsets of a PTBox.

Definition 5.7 (Maximal Satisfiable Subset)**.** *Given a PTBox $PT = (\mathcal{T}, \mathcal{P})$, a subset $\mathcal{P}' \subseteq \mathcal{P}$ is called a **maximal satisfiable subset** of $\mathcal{P}$ if i) $(\mathcal{T}, \mathcal{P}')$ is satisfiable and ii) for any $\mathcal{P}'' \supset \mathcal{P}'$ PTBox $(\mathcal{T}, \mathcal{P}')$ is unsatisfiable. The set of all maximal satisfiable subsets of PT is denoted as $MSS(PT)$.*

Theorem 5.4. *Given a PTBox $PT = (\mathcal{T}, \mathcal{P})$, $MSS(PT) = \{\mathcal{P} \setminus h \mid h \in \mathcal{HS}(MC(PT))\}$ where $\mathcal{HS}(\mathcal{C})$ denotes the set of minimal hitting sets over the set $\mathcal{C}$.*

Proof. We first show that for each $\mathcal{P}' \in MSS(PT)$, $(\mathcal{T}, \mathcal{P}')$ is satisfiable. Assume it is not. Then $(\mathcal{T}, \mathcal{P}')$ contains a conflict which is not in $MC(PT)$. This is contradiction since $MC(PT)$ is complete by Definition 5.4.

Next we show that each element $\mathcal{P}' \in MSS(PT)$ is maximal. Assume it is not. Consider $h = \mathcal{P} \setminus \mathcal{P}'$. If $\mathcal{P}'$ is not maximal w.r.t. set inclusion then h must not be minimal w.r.t. set inclusion which the way $MSS(PT)$ is defined.

Finally we show that $MSS(PT)$ is complete. Let $\mathcal{P}'$ be a subset of $\mathcal{P}$ s.t. $(\mathcal{T}, \mathcal{P})$ is satisfiable and $\mathcal{P}' \nsubseteq \mathcal{P}''$ for all $\mathcal{P}'' \in MSS(PT)$. Consider $h = \mathcal{P} \setminus \mathcal{P}'$. It is not a minimal hitting set over $MC(PT)$ nor a superset of such (otherwise $\mathcal{P}'$ would be a subset of some $\mathcal{P}'' \in MSS(PT)$). Therefore there exists $H \in MC(PT)$ s.t. $h \cap H = \emptyset$ (i.e. a conflict that is not repaired by h). As such, H must be a subset of $\mathcal{P}'$ so $(\mathcal{T}, \mathcal{P}')$ cannot be satisfiable. □

Due to Theorem 5.4, an algorithm for computing $MSS(PT)$ is a straightforward application of Algorithm 6 for computing all minimal conflicts. Its correctness, completeness, and termination all follow from the corresponding properties of Algorithm 6. Note, however, that sometimes it is required to compute the set of satisfiable fragments of a PTBox that are maximal w.r.t. cardinality rather than set inclusion. This is required, for example, in Phase I of the TLexEnt algorithm (see Section 5.4) for computing lex-minimal subsets. We use $MSS_{car}(PT)$ to denote the set of such fragments of PT. It is easy to see that $MSS_{car}(PT) \subseteq MSS(PT)$ so one can get the former from the latter simply by checking cardinality.

5.3 Probabilistic Consistency Algorithms

This section describes two algorithms for solving the PTCon reasoning problem: an optimized version of Lukasiewicz's original algorithm from Section 2.3.3 and a new algorithm based on reduction the Diagnosis problem described in the previous section. We have implemented both algorithms and some performance/scalability evaluation results can be found in Section 7.2.3.

5.3.1 Optimized Original Algorithm

Recall from Section 2.3.3 that the original PTCon algorithm works by computing subsets of the z-partition in a sequential manner. It proceeds from the most general subset, i.e. composed of constraints that are tolerated by all other constraints in the PTBox, to the most specific subset, i.e. composed of constraints that are not tolerated by at least one constraints in all previous subsets. Each next subset is constructed by testing tolerability of *each* remaining constraint by the set of all remaining constraints.

Our optimization is based on the observation that at every step tolerability of *all* constraints sharing the same evidence concept can be tested simultaneously. This observation is formalized in the following lemma:

Lemma 5.1. *Let $(\mathcal{T}, \mathcal{P})$ be a PTBox and $\mathcal{P}$ contains conditional constraints $\phi = (D_1|C)[l_1, u_1]$ and $\psi = (D_2|C)[l_2, u_2]$. Then ϕ is tolerated by $\mathcal{P}$ under $\mathcal{T}$ iff ψ is tolerated by $\mathcal{P}$ under $\mathcal{T}$.*

Proof. By definition ϕ is tolerated by $\mathcal{P}$ under $\mathcal{T}$ if $(\mathcal{T}, \mathcal{P} \cup \{(C|\top)[1, 1]\})$ is satisfiable. Since ϕ and ψ both have C as the evidence concept, ψ is also tolerated by $\mathcal{P}$ under $\mathcal{T}$. □

Consequently, conditional constraints can be grouped by their evidence concepts thus reducing a number of tolerability tests to a single PSAT check. The modified version of the algorithm is shown below. For a set of conditional constraints H we use H_C to denote its subset in which all constraints have the same evidence concept C. Note that the algorithm differs from Algorithm 1 only in line 6.

Algorithm 7: Optimized PTBox consistency algorithm

Input: PTBox $PT = (\mathcal{T}, \mathcal{P})$
Output: z-partition $(\mathcal{P}_0, \ldots, \mathcal{P}_k)$ of $\mathcal{P}$ or *null*

1 **if** $\mathcal{T}$ *is unsatisfiable* **then return** *null*
2 **if** $\mathcal{P} = \emptyset$ **then return** $\emptyset$
3 $H \leftarrow \mathcal{P}$, $i \leftarrow -1$
4 **repeat**
5 $\quad i \leftarrow i + 1$
6 $\quad \mathcal{P}_i \leftarrow \{H_C \subseteq H | (\mathcal{T}, H_C \cup \{(C|\top)[1, 1]\})$ is satisfiable$\}$
7 $\quad H \leftarrow H \setminus \mathcal{P}_i$
8 **until** $H = \emptyset$ *or* $\mathcal{P}_i = \emptyset$;
9 **if** $H = \emptyset$ **then return** $\{\mathcal{P}_0, \ldots, \mathcal{P}_i\}$ **else return** *null*

In the worst case Algorithm 7 is `N2ExpTime`-complete for P-$\mathcal{SROIQ}$ but requires $O({N_E}^2)$ PSAT checks as opposed to $O(N^2)$ for the original algorithm where N is the number of constraints and N_E is the number of distinct evidence concepts (obviously $N_E \leq N$). This could be a dramatic improvement for realistic KBs where relatively few concepts serve as evidences for a large number of constraints. Such scenario is reasonable to expect from medical diagnosis systems where some common symptoms can be probabilistically associated with numerous diseases. Section 7.2.3 supports this expectation using the CADIAG-2 KB as an example.

5.3.2 Diagnosis-driven Algorithm

The second PTCon algorithm is based on solving the Diagnosis problem for a given PTBox. The Diagnosis algorithm presented in previous Section can be used to compute a data structure which stores all information that is essential for constructing the z-partition, if it exists. That structure, which we call a *conflict graph*, defines a mapping between conditional constraints of a PTBox $(\mathcal{T}, \mathcal{P})$ and minimal subsets of $\mathcal{P}$ which do not tolerate them. More formally:

Definition 5.8 (Conflicts and Conflict Graphs)**.** *Given a PTBox $PT = (\mathcal{T}, \mathcal{P})$, the* ***conflict graph*** *$\mathcal{G}(PT)$ is a bipartite graph $(U, \{V_u\}_{u \in U}, E)$ where U is the set of all conditional constraints (i.e. $\mathcal{P}$), and for every $u \in U$, V_u is the set of all minimal subsets of $\mathcal{P}$ that do not tolerate it under $\mathcal{T}$, and $(u, v) \in E$ iff $v \in V_u$. Sets $V_u \cap \{u\}$ are called contextual conflicts under $\mathcal{T}$ or just conflicts.*

The conflict graph is unique for a PTBox. It can be constructed by solving a linear number of the Diagnosis problem instances. More precisely, for each conditional constraint $u = (D|C)[l, u]$ in a PTBox $(\mathcal{T}, \mathcal{P})$, the set V_u can be computed by solving the diagnosis problem for the PTBox $(\mathcal{T}, \mathcal{P} \cup \{(C|T)[1, 1]\})$. This can be done independently for all constraints in $\mathcal{P}$.

Having constructed a conflict graph it is straightforward to turn it into a z-partition or demonstrate that it is not possible. Algorithm 8 is shown below. Similarly to Algorithm 1 we use $\mathcal{P}_C$ to denote the set of all conditional constraints in $\mathcal{P}$ with C as the evidence concept.

As Algorithm 8 shows, consistency of a PTBox can be checked by first constructing the conflict graph, and second, attempting to construct the z-partiton on its basis. If that attempt is successful, then the PTBox is consistent, otherwise it is not. Note that the conflict graph always exists even when the z-partition does not. We next prove its soundness, correctness and termination.

Theorem 5.5. *Algorithm 8 is a sound, complete, and terminating* PTCon *algorithm.*

Proof. Termination The algorithm solves a finite number of the Diagnosis problem instances using the terminating Algorithm 6. Constructing the z-partition from the conflict graph requires the $|U|$ iterations over the graph.

Soundness We prove that ordered partitions computed by the algorithm are z-partitions. By Definition 2.15 a partition $(\mathcal{P}_0, \ldots, \mathcal{P}_k)$ is the z-partition for $(\mathcal{T}, \mathcal{P})$ iff for every $i \in \{0, \ldots, k\}$, $\mathcal{P}_i$ is the set of all constraints from $\mathcal{P} \setminus \bigcup_{j=0}^{i-1} \mathcal{P}_j$ which are tolerated by $H_i = (\mathcal{P} \setminus \bigcup_{j=0}^{i-1} \mathcal{P}_j)$ under $\mathcal{T}$ (note that H_i corresponds to H at i^{th} iteration, line 13). We fix some subset $\mathcal{P}_i$.

First, we show that $(u = (D|C)[l, u]) \in \mathcal{P}_i$ implies ϕ is tolerated by H_i under $\mathcal{T}$. Let $\{V_u\}$ be the set of all minimal subsets of $\mathcal{P} \cup \{(C|\top)[1, 1]\}$ that are unsatisfiable w.r.t

Algorithm 8: Diagnosis-driven PTCon algorithm

Input: PTBox $PT = (\mathcal{T}, \mathcal{P})$
Output: z-partition $(\mathcal{P}_0, \ldots, \mathcal{P}_k)$ of $\mathcal{P}$ or *null*

1 **if** $\mathcal{T}$ *is unsatisfiable* **then return** *null*
2 **if** $\mathcal{P} = \emptyset$ **then return** $\emptyset$
`/* Computing conflict graph` $\mathcal{G} = (U, \{V_u\})$ `*/`
3 $(U, \{V_u\}) \leftarrow (\emptyset, \emptyset)$
4 **for** $C \in \{C|(D|C)[l,u] \in \mathcal{P}$ *for some* $D\}$ **do**
5 $\quad U \leftarrow U \cup \mathcal{P}_C$
6 $\quad$ **for** $u \in \mathcal{P}_C$ **do**
7 $\quad\quad V_u \leftarrow Diagnosis((\mathcal{T}, \mathcal{P} \cup \{(C|\top)[1,1]\}))$
8 $\quad$ **end**
9 **end**
`/* Transforming conflict graph into z-partition */`
10 $H \leftarrow \mathcal{P}$, $i \leftarrow -1$
11 **repeat**
12 $\quad i \leftarrow i + 1$
13 $\quad \mathcal{P}_i \leftarrow \{(D|C)[l,u] \in H | V \nsubseteq H \cup \{(C|\top)[1,1]\}$, for all $V \in \{V_{(D|C)[l,u]}\}\}$
14 $\quad H \leftarrow H \setminus \mathcal{P}_i$
15 **until** $H = \emptyset$ *or* $\mathcal{P}_i = \emptyset$;
16 **if** $H = \emptyset$ **then return** $\{\mathcal{P}_0, \ldots, \mathcal{P}_i\}$ **else return** *null*

$\mathcal{T}$. If u is not tolerated by $\mathcal{F}_i$ then there exists $V \in \{V_u\}$ s.t. $V \in H_i \cup \{(C|\top)[1,1]\}$. Therefore u could not have been added to $\mathcal{P}_i$ at line 13.

Second, we prove the other direction. If u is tolerated by $\mathcal{H}_i$ under $\mathcal{T}$ then $(\mathcal{T}, H_i \cup \{(C|\top)[1,1]\})$ is satisfiable, therefore no set in $\{V_u\}$ is a subset of $H_i \cup \{(C|\top)[1,1]\}$. Consequently, u must be included in $\mathcal{P}_i$ at line 13.

Completeness For completeness we must show that the algorithm generates the z-partition for every consistent PTBox. Assume the algorithm output *null* for a satisfiable PTBox. This is only possible when the structure of the conflict graph is s.t. $\mathcal{P}_i = \emptyset$ on some iteration at line 13. As before we denote the current set H as H_i. Then for all $(u = (D|C)[l,u]) \in H_i$ there exists $V \in \{V_u\}$ s.t. $V \subseteq H_i \cup \{(C|\top)[1,1]\}$ which implies that $(\mathcal{T}, H_i)$ by itself is inconsistent. Probabilistic consistency is a monotonic notion, so from $H_i \subseteq \mathcal{P}$ it follows that $(\mathcal{T}, \mathcal{P})$ is inconsistent, which is a contradiction. □

Algorithm 8 is `N2ExpTime`-complete for P-$\mathcal{SROIQ}$, i.e., has the same worst-case complexity as the original PTCon algorithm and Algorithm 7 which construct z-partitions directly. The follows from the complexity of constructing conflict graphs which is reducible to a linear number of the Diagnosis problem instances (see Section 5.2.1). Although the worst-case complexity is the same, Algorithm 8 involves some overhead which can be quite noticeable in practice: finding all conflicts in a set of conditional constraints might require an exponential number of PSAT checks while the

direct algorithms require only a quadratic numbers of such tests. However, there are certain considerations which can in certain cases outweigh this overhead:

- As explained in Section 5.2.1 the PSAT checks performed when solving the Diagnosis problem can share intermediate results, e.g. generated columns. Therefore they are done considerably faster than if they were performed independently.

- A conflict graph is a richer source of information than a z-partition and can be used for wider purpose. In particular, it provides a better insight inside the PTBox by specifying *precisely* why certain conditional constraints are treated as more specific than others. This enables using conflict graphs as parts of probabilistic explanations, i.e. to explain the user how a certain probabilistic entailment has been computed (including inconsistency).

- A conflict graph greatly facilitates incremental updates to the z-partition. With the original approach, the z-partition needs to be re-computed after every change in the PTBox, regardless of how local that change was. In contrast, conflict graphs are *monotonic* in the sense that they can only grow as new conditional constraints are added in the PTBox. This means that after adding a new statement the graph will retain all of its previous nodes. Furthermore, new minimal non-tolerating sets of constraints can be computed in the context of the existing graph. In particular, when solving the Diagnosis problem it is possible to kickstart from an existing set of conflicts, which will speed up the process. Finally, if a conditional constraint is deleted from the PTBox, then it is necessary and sufficient to remove from the graph all those non-tolerating subsets which contain it. This is because each such subset is minimal according to Definition 5.8.

Taking these considerations into account, we do not claim that one algorithm is strictly superior to another. Our position is that both can be useful depending on usage scenario. For example, Algorithm 7 turned out to be more effective during the PTCon evaluation experiments described in Section 7.2.3 because those PTBox were rich in conflicts (which complicates their diagnosis) but had relatively few evidence concepts. In other situations, for example, involving numerous TLexEnt tasks, Algorithm 8 could well be beneficial.

Finally, consistency of a probabilistic knowledge $(\mathcal{T}, \mathcal{P}, (\mathcal{P})_o)$ base is still decided by first checking consistency of the PTBox and then checking PSAT of $(\mathcal{T}, \mathcal{P}_o)$ for each probabilistic individual $o \in N_{\mathcal{P}}$.

5.4 Tight Lexicographic Entailment Algorithm

This section describes the final advantage of having the diagnosis algorithm, which rectifies the problem of solving too many PSAT instances when computing tight lexicographic entailment with the original method (see Section 2.3.3).

Recall from Section 2.3.3 that the hardest part of non-monotonic lexicographic reasoning is so called Phase I or computation of lex-minimal subsets of a given PTBox $PT = (\mathcal{T}, \mathcal{P})$. According to Definition 2.19 they are unions of subsets of $\mathcal{P}$ that are *maximally satisfiable* w.r.t. an external set of constraints $\mathcal{F}$ and $(C|\top)[1,1]$ (if $(D|C)[l,u]$) is being entailed). Therefore it is possible to apply algorithms for finding maximal satisfiable subsets of $\mathcal{P}$ which are reducible to the Diagnosis problem as explained in Section 5.2.2. Algorithm 9 presents the pseudo-code of the procedure.

Algorithm 9 starts with preliminary checks in lines 1–2 to correctly deal with the situations when either the PTBox is inconsistent (in which case the meaning of lex-minimal subsets is not well-defined) or the set $\mathcal{F} \cup \{(C|\top)[1,1]\}$ contradicts the TBox. The latter occurs in case of PKB inconsistency for a certain probabilistic individual. The remainder of the algorithm is split onto two main phases: Phase I (computing lex-minimal subsets of $\mathcal{P}$) and Phase II (tight logical entailment from lex-minimal subsets). Phase II is equivalent to the original version while Phase I is novel and is explained in more detail.

Phase I is largely enclosed in the **for** loop between lines 6 and 26. The algorithm iterates over the previously computed z-partition and incrementally computes lex-minimal subsets. The first and the most important part of the loop is lines 8–16. The current set of lex-minimal subsets in stored in $\mathcal{LM}$ which is initialized prior to the loop at line 4. The algorithm tries to extend each subset $L \in \mathcal{LM}$ by including some maximal subsets of $\mathcal{P}_i$ which are *not* in conflict with it. In other words, the task is reduced to finding maximal satisfiable fragments of a set of conditional constraints $\mathcal{P}_i$ given the TBox $\mathcal{T}$ and some other set of conditional constraints L. This is a direct application of our Diagnosis algorithm, as explained in Section 5.2.2, which is invoked in line 9. Note that elements of R are maximal w.r.t. cardinality as opposed to set inclusion. Next, the algorithm uses an auxiliary associative array $\mathcal{H}$ to associate each $L \in \mathcal{LM}$ with the maximal subsets of $\mathcal{P}_i$ (i.e. the *extensions* of L to the next subset in the z-partition). The second part of Phase I (lines 17–25) is about updating the current lex-minimal subsets using the data computed in the previous loop and stored in $\mathcal{H}$. Note that the current lex-minimal subsets are discarded when a new, lex-preferable subset is found (line 12). Finally, observe that the algorithm retains the possibility of approximate entailment featured by the Lukasiewicz's algorithm. If the MSS_{car} sub-procedure returns a subset of maximal satisfiable fragments if $\mathcal{P}_i$ (for example, by skipping some ways of repairing unsatisfiability of $(\mathcal{T}, \mathcal{P}_i \cup L)$) then the resulting interval will be a superset of

Algorithm 9: Diagnosis-driven TLexEnt algorithm

Input: PTBox $PT = (\mathcal{T}, \mathcal{P})$, finite set of conditional constraints $\mathcal{F}$, basic concepts C, D

Output: $(l, u) \in [0, 1]^2$ such that $\mathcal{F} \models^{lex}_{tight} (D|C)[l, u]$ under PT

`/* Preliminary checks */`

if *PT is inconsistent* **then return** *(1,0)*

if $(\mathcal{T}, \mathcal{F} \cup \{(C|\top)[1, 1]\})$ *is unsatisfiable* **then return** *(1,0)*

`/* Phase I: computing lex-minimal subsets */`

$(\mathcal{P}_0, \ldots, \mathcal{P}_k) \leftarrow$ z-partition of PT

$\mathcal{LM} \leftarrow \{\mathcal{F} \cup \{(C|\top)[1, 1]\}\}$

$c \leftarrow 1$

for $i \leftarrow k$ **to** 0 **do**

  $\mathcal{H} \leftarrow \emptyset$

  for $L \in \mathcal{LM}$ **do**

    $\mathcal{R} \leftarrow MSS_{car}((\mathcal{T}, \mathcal{P}_i), L)$

    $r \leftarrow$ cardinality of sets in $\mathcal{R}$

    if $r \geq c$ **then**

      if $r \neq c$ **then** $\mathcal{H} \leftarrow \emptyset$

      $\mathcal{H} \leftarrow \mathcal{H} \cup (L, \mathcal{R})$

      $c \leftarrow r$

    end

  end

  $\mathcal{LM}_{new} \leftarrow \emptyset$

  for $L \in \mathcal{LM}$ **do**

    if $(L, \mathcal{R}) \in \mathcal{H}$ *for some* $\mathcal{R}$ **then**

      for $R \in \mathcal{R}$ **do**

        $\mathcal{LM}_{new} \leftarrow \mathcal{LM} \cup \{L \cup R\}$

      end

    end

  end

  $\mathcal{LM} \leftarrow \mathcal{LM}_{new}$

end

`/* Phase II: TLogEnt from lex-minimal subsets */`

$(l, u) \leftarrow (1, 0)$

for $L \in \mathcal{LM}$ **do**

  compute $c, d \in [0, 1]$ s.t. $(\mathcal{T}, L) \models_{tight} (D|\top)[c, d]$

  $(l, u) \leftarrow (min(l, c), max(u, d))$

end

return (l, u);

the exact interval.

Theorem 5.6. *Algorithm 9 is a correct and terminating* TLexEnt *algorithm.*

Proof. Termination Follows from termination of the MSS_{car} and TLogEnt algorithms. The former is invoked k times, where k is the number of subsets in the z-partition, and the letter is invoked m times, where m is the number of lex-minimal subsets.

Correctness It suffices to prove that at the end of Phase I $\mathcal{LM}$ indeed stores lex-minimal subsets. Assume otherwise, that is, there exists a set $R = R_0 \cup \cdots \cup R_k$ where $R_i \subseteq \mathcal{P}_i$ s.t. i) $(\mathcal{T}, R \cup \mathcal{F} \cup \{(C|\top)[1,1]\})$ is satisfiable and ii) L is lex-preferable to every $L \in \mathcal{LM}$. The second condition means that there exists $i \in \{0, \ldots, k\}$ s.t. $|R \cap \mathcal{P}_i| > |L \cap \mathcal{P}_i|$ and $|R \cap \mathcal{P}_j| = |L \cap \mathcal{P}_j|$ for every $j = \{i+1, \ldots, k\}$ and every $L \in \mathcal{LM}$. Now consider the i^{th} iteration of Phase I inner loop (lines 8–15). Assume w.l.o.g. that $\mathcal{LM}$ consisted of a single subset $L^{(i)} = L_0 \cup \cdots \cup L_i$ at that point. There are only three cases which could eventually lead to missing R in $\mathcal{LM}$:

- R_i was computed in line 9 *before* L_i and then discarded in line 12. This is not possible because $|R_i| > |L_i|$, so $c > r$ (line 11).
- R_i was computed in line 9 *after* L_i but not added as an extension of some current lex-minimal subset (i.e. lines 11–15 were not executed). This is not possible for exactly the same reason.
- R_i was *not* computed in line 9 at all. However, $(\mathcal{T}, R^{(i)})$ is satisfiable (follows from i)) and $|R_i| > |L_i|$ so this case is also not possible provided the algorithm for computing MSS_{car} is correct (i.e. it cannot compute $|L_i|$ but not $|R_i|$).

Therefore, such R does not exists, which completes the proof. □

In contrast to the original TLexEnt algorithm, Algorithm 9 presents a diagnosis-driven, goal directed approach to computing lex-minimal subsets. Pinpointing conflicts enables quicker computation of maximal satisfiable subsets than an uninformed search, especially if contextual conflicts overlap. This process will be referred to *conflict resolution* in later sections.

Consider the following example which illustrates the advantages of Algorithm 9 in comparison to the original algorithm.

Example 5.3. *Let* $(\mathcal{T}, \mathcal{P})$ *be the following PTBox:*
$\mathcal{T} = \{S_3 \sqsubseteq S_2, S_3 \sqsubseteq S_1\} \cup \bigcup_{i \in \{1,\ldots,N\}} \{D_i \sqsubseteq \top\}_i$
$\mathcal{P} = \{(D_1|S_1)[0.9, 1.0], (D_1|S_2)[0.8, 0.9], (D_1|S_3)[0, 0.1]\} \cup \bigcup_j \{(D_j|S_1)[l_j, u_j]\}_j$
where $j \in \{2, \ldots, N\}$ *and* $[l_j, u_j]$ *are arbitrary sub-intervals of* $[0, 1]$.

Let N be some large number such as 100, and assume we are interested in solving TLexEnt *for $(D_j|S_3)$ for some j. The z-partition of the PTBox has the following form: $(\{(D_1|S_1)[0.9,1.0],(D_1|S_2)[0.8,0.9]\} \cup \bigcup_j \{(D_j|S_1)[l_j,u_j]\}, \{(D_1|S_3)[0,0.1]\})$ because $(D_1|S_3)[0,0.1]$ is the only constraint which is not tolerated by the rest of $\mathcal{P}$.*

We first consider the behavior of the original algorithm. Since there are only two subsets in the z-partition and $\{(S_3|\top)[1,1]\}$ (the initial lex-minimal subset) is trivially satisfiable with $\mathcal{P}_1$, the algorithm quickly proceeds to lines 10–18 (see Algorithm 2), i.e. finding the maximal subsets of $\mathcal{P}_0$ which are satisfiable with $\mathcal{P}_1 \cup \{(S_3|\top)[1,1]\}$. It performs a binary search on $\mathcal{P}_0$ solving PSAT *for each subset of progressively decreased size (line 14). The size of $\mathcal{P}_0$ is 99, so in the worst case line 14 may require C^{45}_{99}* PSAT *tests. This is the case even if the algorithm is optimized to move to checking subsets of the next size as soon as K' is non-empty because one of the non-tolerating constraints $(D_1|S_1)[0.9,1.0],(D_1|S_2)[0.8,0.9]$ can be a part of some $G \subseteq \mathcal{P}_0$ being checked until the very end (i.e. the search is blinded).*

Algorithm 9 processes $\mathcal{P}_0$ very differently. It invokes the MSS algorithm which requires only three PSAT *checks to pinpoint two minimal unsatisfiable subsets: $\mathcal{C}_1 = \{D_1|S_1)[0.9,1.0],(D_1|S_3)[0,0.1],(S_3|\top)[1,1]\}$ and $\mathcal{C}_2 = \{D_1|S_2)[0.8,0.9]$, $(D_1|S_3)[0,0.1]$, $(S_3|\top)[1,1]\}$.*[12] *The first test is needed to discover $\mathcal{C}_1$, the second is needed to discover $\mathcal{C}_2$ after excluding $\{D_1|S_1)[0.9,1.0]\}$ from $\mathcal{P}_0$, and the final is needed to exclude both $\{D_1|S_1)[0.9,1.0]\}$ and $\{D_1|S_2)[0.8,0.9]\}$ to prove that no other conflicts exist. Once that has been done the set of maximal satisfiable fragments of $\mathcal{P}_0$ can easily be constructed (it is a dual of $\{\mathcal{C}_1,\mathcal{C}_2\}$).*

The difference in the number of PSAT tests illustrated by the Example 5.3 is due to the fact that the PTBox contains a large number of conditional constraints $\{(D_j|S_1)[l_j,u_j]\}_j$ that had nothing to do with S_3. The only constraints that do not tolerate $D_1|S_3)[0,0.1]$ are $(D_1|S_1)[0.9,1.0],(D_1|S_2)[0.8,0.9]$, but the original algorithm cannot figure that out thus having to resort to the blinded search over the powerset of $\mathcal{P}_0$. On the other hand our algorithm can efficiently filter out the irrelevant parts and compute the conflicts.

This example is not purely artificial. For example, in the CADIAG-2 case, concepts S_1, S_2, S_3 can stand for symptoms and D_i for diseases (see Section 3.2). One of S_1, S_2 can be one of common symptoms probabilistically associated with numerous diseases thus producing the constraints $\{(D_j|S_1)[l_j,u_j]\}_j$ which have nothing to do with S_3. Algorithm 9 is extremely efficient in handling such cases when the number of conflicts (or, more precisely, the number of ways to repair them) is relatively low. The number of required PSAT tests will grow exponentially with the number of repairs, so it is

[12] Pinpointing each unsatisfiable subset is more than just a PSAT test (see Section 5.2.1) as it also involves the minimization step, but the difference is negligible.

not hard to imagine a situation when it becomes infeasible to compute lex-minimal subsets (see evaluation experiments in Section 7.2.4). Fortunately, as our experience with CADIAG-2 shows, such situations seem to be far less common than those similar to Example 5.3.

Also there are additional advantages of Algorithm 9, similar to those listed in Section 5.3.2. Below we describe one important possibility, namely maintaining a conflict graph for each probabilistic individual which is important for efficiency of PABox entailments.

PABox entailments are quite an important use case, for example they can be used to compute medical diagnoses for patients whose medical data is represented as a collection of PABox constraints. Consequently it is desirable to maintain a data structure which would speed up computation of lex-minimal subsets w.r.t. constraints in the PABox. If all constraints in the PABox are unconditional then one may pre-compute the set of all lex-minimal subsets to be used for all subsequent entailments.

Using conflict graphs helps to take this idea even further. Given a PTBox $(\mathcal{T}, \mathcal{P})$ and a PABox $\mathcal{P}_o$ one may maintain a conflict graph for $(\mathcal{T}, \mathcal{P} \cup \mathcal{P}_o)$. While this requires computing lex-minimal subsets from the conflict graph for each entailment, the benefits are substantial. First, the conflict graph provides a great insight into how probabilistic knowledge about the individual interacts with the generic PTBox axioms. Second, any changes in $\mathcal{P}_o$ can be propagated into the conflict graph without recomputing the whole structure.

Chapter 6

Synthetic Performance Evaluation

Performance evaluation of the main reasoning algorithms has been split into two parts: synthetic evaluation (presented in this chapter) and application-based evaluation (presented in the next chapter). Synthetic evaluation procedures, which use artificially generated data, have been carried out for the probabilistic satisfiability algorithm. Evaluation of the Diagnosis and the TLexEnt algorithms is different from PSAT in the sense that their performance critically depends on the number and overlap of contextual conflicts in a probabilistic knowledge base (as explained in sections 5.2 and 5.4). Since very few probabilistic KBs already exist it is difficult to synthesize random test instances whose conflict space would usefully model realistic scenarios. Consequently, we have opted to use existing KBs, namely fragments of CADIAG-2 to carry out evaluation of the Diagnosis and TLexEnt algorithms and deferred it to Chapter 7.

This chapter describes the main factors on PSAT performance that we investigate, as well problem instance generation methodologies and evaluation environment. It also discusses the results of Pronto's effectiveness on various classes of input. The evaluation procedures pursue three general goals:

- Scalability, performance and robustness *evaluation*. The procedures aim to test Pronto's performance on progressively large or hard KBs and measure any substantial variability in the main performance metrics.

- Scalability and performance *understanding*. The procedures should provide insight into the sources of the computational complexity of the reasoning procedures in P-$\mathcal{SROIQ}$ and the implementation of the PSAT algorithm.

- Correctness testing. The procedures aim to check correctness of Pronto's PSAT algorithm by testing it on KBs which have been generated in such a way that

their satisfiability is known upfront.

The procedures are *not* divided into categories according to the goals. The reason is that those problem instances, which are most reasonably used to prove correctness, such as satisfiable knowledge bases, also appear to be the hardest and the most informative from the computational perspective. Therefore, the same procedures are used to evaluate scalability and performance while the results are simultaneously verified for correctness.

The chapter is structured as follows. First, Section 6.1 describes the evaluation methodology, in particular, the algorithms for generating various classes of test knowledge bases. Following a brief description of the evaluation environment sections 6.3, 6.4, and 6.5 present the tests that evaluate the performance, scalability and robustness of the PSAT algorithm on propositional knowledge bases, approximate translations of Bayesian networks, and probabilistic extensions of real OWL ontologies, respectively. Finally, Section 6.6 concludes the chapter with the summary of the obtained results.

6.1 Generic Evaluation Methodology

The evaluation methodology consists of algorithms for generating test instances of the PSAT problem, metrics used to assess the performance, and means of gathering the required measures while running experiments.

6.1.1 Test Data Parameters

The first step in designing the evaluation experiments is to define the set of characteristics of P-$\mathcal{SROIQ}$ knowledge bases whose impact on performance of the reasoning algorithms needs to be measured. Such characteristics are called *complexity factors* below.

The theoretical insight into some of the major complexity factors for the PSAT algorithm was presented in Section 5.1.4. They include the size of probabilistic signature, the size of the PTBox, and the richness of the TBox. The first two factors are straightforward parameters which can easily be varied when generating a random PTBox. In addition we experiment with the proportion of unconditional constraints in the PTBox. Although this parameter is not expected to influence performance of the possible world generation algorithm, it may have an impact on convergence as well as on hardness of the master linear program (5.5).

TBox richness is not straightforward since i) it may not be possible to compute it for a given TBox in reasonable time since the number of entailments can be exponential, and ii) it is not clear how to generate a TBox with a specified richness. In this thesis we use another metric which approximates richness and which can be computed

using a polynomial (in the size of signature) number of SAT tests in the target DL. The metric, which we call *subsumption density*, represents the number of subsumption axioms entailed by a TBox $\mathcal{T}$ which connect concepts, or their negations, from the probabilistic signature. Any such subsumption effectively reduces the total number of possible worlds by 25%.[1] We give the formal definition below:

Definition 6.1 (Subsumption Density)**.** *Subsumption density $\mathcal{D}$ of a PTBox $(\mathcal{T}, \mathcal{P})$ with probabilistic signature Φ is the ratio $\frac{S_{\mathcal{T}}(\Phi)}{|\Phi|}$, where $S_{\mathcal{T}}(\Phi)$ is the total number of subsumptions $C \sqsubseteq D$ entailed by $\mathcal{T}$, where both C and D are concept expressions from Φ or their negations.*

There are two important features of the density function. First, it counts the number of literal, not atomic, subsumptions. As such it also involves disjointness. Second, it counts *inferred* subsumptions (asserted axioms are trivially inferred), in particular, takes into account transitivity of the subsumption relationship.

Obviously the density metric is only an approximation of TBox richness (see Definition 5.3). It does not take into account any entailments other than binary subsumptions. As such it may not be accurate for knowledge bases, for example, those containing numerous Horn expressions of the form $C_1 \sqcap \cdots \sqcap C_k \sqsubseteq D$ where $k > 2$. We leave it for a future work to investigate whether a finer approximation is computationally feasible.

However, the metric has few advantages. First, it is relatively easy to compute for a given TBox. This helps selecting real ontologies such that they substantially differ in density. Second, it is relatively straightforward to generate random concept hierarchies with given subsumption density. This helps to measure performance of the PSAT algorithm as a function of density. Finally, in spite of being possibly inaccurate on some TBoxes, the metric does allow us to control size of the MILP program (5.4) used to generate columns. This is so because each literal subsumption counted by the metric needs to be represented as a linear inequality in the program.

Summing up, the following parameters will be varied when generating test PTBoxes to evaluate the PSAT algorithm:

- PTBox size (the total number of conditional and unconditional constraints),
- probabilistic signature size (the total number of basic probabilistic concepts appearing in some conditional constraints in PTBox),
- proportion of unconditional constraints in PTBox,
- subsumption density of TBox (with respect to the probabilistic signature).

[1] Let A, B be concepts in Φ. The set of all possible (w.r.t. $\mathcal{T}$) worlds over Φ can be partitioned onto four sets of equal size: $\mathcal{I}_{A,B}$, $\mathcal{I}_{A,\neg B}$, $\mathcal{I}_{\neg A,B}$, and $\mathcal{I}_{\neg A,\neg B}$ which contain worlds with literals $\{A, B\}$, $\{A, \neg B\}$, $\{\neg A, B\}$, $\{\neg A, \neg B\}$ respectively. If the $\mathcal{T}$ entails $A \sqsubseteq B$ then the set $\mathcal{I}_{A,\neg B}$ is empty.

Although we do not aim at a full exploration of the entire parameter space, below we systematically try various realistic combinations of parameter values to provide a reasonably complete picture of PSAT performance, scalability, and robustness.

6.1.2 Test Data Generation

The central component of the evaluation methodology is the generation of test problem instances. Unless explicitly stated otherwise, the problem type is assumed to be PSAT, so it is sufficient to generate a PTBox only. This is highly non-trivial for two reasons: First, there are neither large and naturally occurring P-$\mathcal{SROIQ}$ ontologies nor established methodologies for modeling in P-$\mathcal{SROIQ}$ which could be used for test data generation. Thus a part of this thesis' contribution is a methodology for generating test knowledge bases. Second, even in the propositional case the structure of the space of all probabilistic knowledge bases is complicated and uniform sampling may easily lead to selecting either unsatisfiable or massively incoherent PTBoxes.

The first problem is fairly typical for many novel formalisms. Large and realistic knowledge bases cannot be expected to exist until the formalism provides a solid modeling and tool support which, in turn, require modeling experience and benchmark suites. We tackle this problem here partly by utilizing our experience from building the BCRA ontology and discovering inconsistencies in the probabilistic formalization of the CADIAG-2 knowledge base. In particular, this experience suggests that realistic P-$\mathcal{SROIQ}$ PTBoxes will mostly contain conditional constraints since they are more adequate means of capturing generic probabilistic relationships than unconditional uncertain implications (see Section 3.1 for more details).

The second problem deserves a more detailed discussion. For the sake of simplicity assume that TBoxes are restricted to collections of propositional clauses and PTBoxes are formed by picking concept names randomly among those that appear in the clauses. Then uniform sampling from the space of all PTBoxes of a size k can be done by first, generating a collection of clauses (the TBox), second, selecting $2 * k$ concept names and third, splitting the concept names into pairs and assigning them a random real interval within $[0, 1]$. This method has two undesirable properties:

- It is strongly biased towards satisfiable PTBoxes.
- It is strongly biased towards incoherent PTBoxes.

The first bias is due to the semantics of P-$\mathcal{SROIQ}$ according to which each conditional constraint $(D|C)[l, u]$ is satisfied by a probabilistic interpretation Pr such that $Pr(C) = 0$ (such *vacuous satisfiability* was discussed in Section 2.3.4). Such vacuous satisfiability can be prevented in several ways, for example, by adding an unconditional constraint $(C|\top)[l, u]$, where $l > 0$, or by adding two constraints $\{(C|A)[l_1, u_1]$,

$(C|\neg A)[l_2, u_2]\}$ and so on. None of these are likely to occur in uniformly sampled conditional PTBoxes.

Unfortunately, uniformly sampled PTBoxes are often satisfiable but incoherent (recall, that incoherent PTBoxes entail zero probability for some of the concepts in their probabilistic signature). In other words, they are not satisfiable in a "useful way", i.e. they do have conflicts between conditional constraints but those are masked by vacuously satisfying probability distributions. The reason for that is the uniform sampling of probability intervals which either contradict each other or are too wide and, consequently, uninteresting.

These issues indicate that uniformly sampled PTBoxes are unlikely to be good representatives of future handcrafted probabilistic knowledge bases, so the evaluation results may not be informative. Consequently some controlled random process of constructing PTBoxes is required. It is important that the process avoids any of the aforementioned pitfalls and ensures a certain diversity of PTBoxes in order to better understand the performance of Pronto's reasoning algorithms and sources of its complexity. Test PTBoxes used in the evaluation differ in the following key characteristics:

- Expressivity of the classical part of the KB. The methodology distinguishes between classical parts expressed in propositional and non-propositional DLs.
- Richness of the classical part measured in terms of its *subsumption density* (see Section 6.1.1, esp. Definition 6.1).
- Method of assigning probability values to conditional constraints. Possible methods are those which ensure (un)satisfiability of the KB, those which assign probabilities at random (see Section 6.1.2), or obtaining the probabilities from external sources, such as ontology alignment tools.
- Method of obtaining the classical part such as generation of random concept hierarchies or use of real-life OWL ontologies.

We next describe the general procedure used to generate test PTBoxes for the evaluation. The procedure is parameterized and involves a number of auxiliary algorithms, for example, for ensuring satisfiability or generating a classical part of the PTBox with specific characteristics.

On a high level the procedure is composed of the following two main steps:

- *Obtaining the classical part of the PTBox.* This provides a signature for the probabilistic part, i.e. conditional constraints.
- *Obtaining the probabilistic part of the PTBox.* This involves generation of both structure of probabilistic statements and probability intervals.

Next we describe the steps in more detail and present the options for obtaining both components of test PTBoxes.

Classical Part of Test PTBoxes

There are three principal ways for obtaining the classical part of test PTBoxes that are used in the experiments below:

- Empty classical part
- Random generation of concept hierarchies (taxonomies)
- Use of existing OWL ontologies from real domains and applications

The first option implies that the TBox in each test PTBox is the empty set of axioms. However, generation of concept names is still required, so the classical part can be regarded as a set of trivial axioms $A_i \sqsubseteq \top$ for each concept name A_i. Such vacuous TBoxes are used in experiments with collections of random unconditional clauses (Section 6.3.1) and in for approximate translations of Bayesian networks (Section 6.4).

For the second method, the classical part of a test PTBox is generated in a random way. It involves generation of the signature as well as TBox axioms to obtain the hierarchy. We only generate propositional concept hierarchies in this evaluation, but it is certainly possible to extend the procedure to random DL TBoxes and OWL ontologies [141]. This method is parameterized to produce TBoxes with required *subsumption density*, which is a metric that reflects the number of subsumption axioms (both asserted and inferred) over a given set of concept names. The details are given in Section 6.3.2.

The third method is used in experiments that evaluate our reasoning algorithms on probabilistic extensions of real ontologies. The details on selection of existing ontologies from the TONES repository[2] are presented in Section 6.5.

Probabilistic Part of Test PTBoxes

Probabilistic parts of test PTBoxes, which are used in this chapter, are produced by a random generation process that takes a probabilistic signature (collection of concept expressions) as an argument. Such processes are composed of the following two steps:

- Selection of concept pairs
- Generation of probability intervals

[2] `http://owl.cs.manchester.ac.uk/repository/`

Observe that a similar approach to constructing probabilistic knowledge bases is used, for example, for generating or learning graphical models such as Bayesian networks. There it is also common to first obtain the structure and then the probabilities.

The first step is generation of the structure of PTBox constraints, i.e., random selection of the evidence and conclusion concepts from the specified signature. Typically the experiments require that proportion of constraints be unconditional. This is due to two reasons. First, each unconditional constraint, e.g. $(C|\top)[l,u]$ where $l > 0$, precludes the vacuous satisfiability of a set of conditional constraints $(D_i|C_i)[l_i,u_i]$ where C is a subconcept of each C_i, because the probability of each C_i has to be non-zero. In the absence of unconditional constraints almost all randomly generated collections of conditional constraints will be satisfiable because most of evidence concept can have zero probability. The second reason is that a small ratio of unconditional constraints reflects a likely common modeling pattern in P-$\mathcal{SROIQ}$. According to our experience with the BCRA ontology, unconditional constraints are often used in a PABox to represent probabilistic facts, or beliefs, about a specific individual (see Section 3.1 and [110]). They play their role during PSAT solving when they are combined with the PTBox during consistency checking or lexicographic reasoning.

Given the structure of the PTBox the rest is to generate probability intervals. Ultimately it is the probability intervals attached to constraints that determine whether the resulting PTBox will be satisfiable or not. Willful generation of satisfiable or unsatisfiable probabilistic KBs is important for several reasons. First, this is the prime method for checking correctness of the satisfiability algorithm. Second, satisfiability is one of the crucial hardness metrics for performance evaluation. It has been reported in several publications that satisfiable KBs are often harder for PSAT algorithms [45, 81], in particular, for those which implement special techniques for early conflict detection, for example, based on probability propagation rules [81]. Finally, satisfiable KBs are required for the evaluation of (non-monotonic) entailment algorithms.

Unfortunately, random assignment of probabilities to generated constraints is likely to result in an unsatisfiable PTBox provided that it contains unconditional statements [98, 45, 81]. Therefore special techniques are required to ensure satisfiability. Two such techniques have been used before. The first is based on the idea of avoiding conflicts between different probabilistic statements analogously to Bayesian networks, i.e., by making sure that local probability distributions always represent a global probability distribution across a loop-free network (such method is first used in [45]). The second is based on generation of probabilistic interpretations which can then be used to assign probabilities to statements [98]. In that case satisfiability is guaranteed because satisfying interpretations (models) have been constructed explicitly.

A generalization of the second technique is used in all experiments below, except for approximate translations of Bayesian networks. Its main advantage is that it works with any probabilistic KB, propositional or not, and does not impose any restrictions on its structure (such as cycle disallowance). For the current evaluation it has been implemented in the following steps: First, two sets of possible worlds $\mathcal{I}^1_\Phi, \mathcal{I}^2_\Phi$ of size $k \geq 2 \times |\mathcal{P}|$ are generated for a PTBox $(\mathcal{T}, \mathcal{P})$ with probabilistic signature Φ. The size of the sets of possible worlds is at least twice the size of $\mathcal{P}$ in order to avoid constructing artificially small[3] solutions to the linear program (2.3). Second, probabilistic interpretations Pr_1, Pr_2 are defined by generating two sequences of k random numbers summing to 1 which represent probabilities of possible worlds in $\mathcal{I}^1_\Phi$ and $\mathcal{I}^2_\Phi$. Third, the lower probability l (resp. the upper probability u) for each constraint $(D|C)[l, u]$ in $\mathcal{P}$ is determined as the smallest (resp. the largest) of values $Pr_i(D|C)$, where $i \in \{1, 2\}$.

A possible world generation algorithm plays the central role in this method. In order to avoid possible bias in evaluation results it has to ensure that all concept names in Φ occur in possible worlds of $\mathcal{I}_\Phi$ with approximately equal frequency. Algorithm 10 ensures that by doing random runs through the powerset of Φ. In each run it uses a $\mathcal{SROIQ}$ reasoner to make sure that every subset being added to the resulting set $\mathcal{I}_\Phi$ is a possible world, i.e., is realizable for a fresh individual given $\mathcal{T}$.

Algorithm 10: Possible world generation algorithm

Input: PTBox $PT = (\mathcal{T}, \mathcal{P})$, $N > 0$
Output: Subset of possible worlds $\mathcal{S}$ of size N

```
Φ ← probabilistic signature of PT;
for i=1 to N do
    I_i ← ⊤;
    foreach C in Φ do
        X ← choose randomly between C and ¬C;
        if T ⊨ X ⊑ ⊥ then
            I_i ← I_i ⊓ ¬X;
        else
            I_i ← I_i ⊓ X;
        end
        S ← S ∪ {I_i};
    end
end
return S;
```

[3]Recall from Section 5.1 that a system of linear inequalities always has a solution in which the number of non-zero variables is no greater than the number of inequalities. It may have fewer variables with non-zero values (a *degenerate* basis). However, generating KBs whose PSAT linear systems *necessarily* have such property is undesirable in the evaluation process because it can be exploited by the reasoning algorithms.

The algorithm makes the assumption that the total number of possible worlds is far greater than N, so the chance of generating the same world twice is negligibly small. This assumption works fine in our experiments since they deal with large signatures of more than a hundred concept names. Also the real implementation is a bit more involved, for example, it exploits the TBox classification results and uses extensive caching to speed up the generation process.

Unsatisfiable knowledge bases are generated in a simpler way where each constraint in $\mathcal{P}$ was assigned a random probability interval. Interval bounds are pseudo random numbers generated from a normal distribution with a mean of $0.4 + alea(0, 0.3)$ where $alea(a, b)$ is a random number between a and b and a standard deviation equal to 0.2 (i.e. similarly to [98]). Such procedure usually, but not always, creates unsatisfiable KBs. To account for an unexpected satisfiability Pronto was asked to provide a model in those rare PSAT correctness tests where it reported that the KB was satisfiable.

We implemented the synthetic test data generation methodology PREVAL-DL—the framework for a systematic evaluation of probabilistic DL reasoners.[4] In addition to KB generation algorithms it provides facilities for running various reasoning procedures in different modes, in particular, using dedicated threads or processes per reasoning task. The framework can be extended to generate KBs in other formalisms apart from P-$\mathcal{SROIQ}$.

6.1.3 Performance Measures and Data Gathering

All of the evaluation tests presented below use wall time as the main measure of performance.[5] The main advantages of wall time are that it is simple to measure and it can be used for all algorithms, regardless of their implementation details, and is easy to interpret. Another possibility is to use CPU time for more precise measurements. However, given that most of the tests take minutes to complete and are executed a number of times the extra precision is not worth the effort.

Wall time is simple, but not sufficiently informative measure, especially when it comes to evaluating scalability. A single number does not provide much insight into the behavior of the algorithm under evaluation, in particular, does not help to reveal "bottlenecks", i.e., pieces of code which take long time or are executed an excessive number of times. For example, CPU time would not tell how often a P-$\mathcal{SROIQ}$ reasoner invoked an underlying $\mathcal{SROIQ}$ reasoner while such calls can very well be one of the bottlenecks. Therefore, most of the tests also use other measures, in particular, the following two:

[4] `http://www.cs.man.ac.uk/~klinovp/research/prevaldl/index.html`

[5] Wall time (or wall clock time) measures real time from the start of an operation to its end, which includes any artificial delays, such as IO operations.

Number of generated columns This metric represents the speed of convergence of a column generation algorithm. It counts only valid columns which have been added into the master linear program during PSAT/TLogEnt procedure. Note that it does *not* necessarily represent the number of simplex pivots since multiple columns can sometimes be added at a single pricing step.

Total column generation time (CG Total) This metric represents the total time the PSAT algorithm spends on generating and validating columns (i.e., it is the sum of run times of all invocations of Algorithm 4).

In addition, one can analyze the difference between the total running time and the total column generation time, which is mostly the time spent on optimizing RMP. This measure is not explicitly present in the tables below because RMP's optimization, however costly at the moment, can hardly be a key obstacle to scalability of the PSAT algorithm since i) it is a polynomial time problem and ii) the results can be improved simply by tuning the RMP's formulation and the LP solver. Both such aspects are beyond the scope of this thesis.

Pronto includes an internal telemetry subsystem which enables the gathering of these performance measures. In particular, the linear program manager (see Section 8.2) tracks each generated column and measures the time it takes to generate them. In addition, the column generator has the ability to count the number of validation tests for each column candidate and expose that number to the linear program manager.

6.2 Evaluation Environment

Pronto is written in Java and compiled using Sun JDK 1.6. All evaluation tests have been performed on a PC with a 2GHz CPU, 2GB of RAM, Sun JRE 1.6.0_07 running under Windows XP SP3. The only JVM option that was used for performance tuning was `-Xmx` to allow the reasoner to use the maximal available amount of memory.

6.3 Random Propositional Knowledge Bases

In our first series of experiments we evaluate the PSAT algorithm on randomly generated propositional knowledge bases (PTBoxes). The full expressivity of P-$\mathcal{SROIQ}$ is not necessary in this case as such PSAT instances can be solved by a propositional PSAT algorithm. However, the experiments are required for the following reasons:

- To compare performance and scalability of the hybrid algorithm to the previous methods which capture the entire structure of propositional knowledge bases in the column generation model.

- To understand how well the hybrid algorithm deals with propositional problems which are an important special case in P-$\mathcal{SROIQ}$.
- To understand how strong the algorithm is influenced by the amount of the classical knowledge in a controlled environment (e.g., the subsumption density can be varied).

The first experiment is conducted on collections of random probabilistic clauses without any classical knowledge and was designed specifically for the comparison. The second experiment answers the next to questions by evaluating the algorithm on random concept hierarchies. There we have the full control over the size and complexity of the classical part as opposed to the experiments with real ontologies. In particular, it is possible to generate propositional TBoxes with the required subsumption density (see Section 6.3.2) so that we can compare performance on PTBoxes which differ *only* in their density.

6.3.1 Random Clauses

PTBoxes used in the first experiment are collections of probabilistic propositional clauses, i.e. unconditional constraints of the form $(C_1 \sqcup \cdots \sqcup C_k|\top)[p,p]$ where each C_i is a literal concept (a concept name or its complement). Such constraints are probabilistic generalizations of implications of the form $C_1 \sqcap \cdots \sqcap C_{k-1} \sqsubseteq C_k$.[6] Since the main aim of this experiment is to compare performance of our algorithm with the previously reported results of propositional solvers, we have followed the methodology for generating random KBs which was described by Jaumard et al. [98] and later used in [77, 81]

Jaumard's methodology easily fits into our generic procedure described above. The first step (obtaining the classical part) amounts to generation of a plain list of concept names of specified size. The number of concept names was kept fixed to 200 since it is the largest number of atoms in [81]. On the second step (obtaining the probabilistic part) we first generated the required number of disjunctive expressions and then assigned probability intervals in the satisfiability preserving way by using Algorithm 10. The length of each expression is a random variable uniformly distributed between 1 and 4. Polarity of each literal is also uniformly random.

The results are presented in Table 6.1. As reported in [81] the previously developed propositional PSAT algorithm on average generated about 3300 columns for collections of 800 clauses over 200 atoms. Our algorithm generates about 10 times fewer columns (we do not compare the total time to abstract away from the hardware differences). Even more importantly, our algorithm does not seem to exhibit a super-polynomial

[6] An implication of the form $C_1 \sqcap \cdots \sqcap C_{k-1} \sqsubseteq C_k$ can be rewritten as $\neg C_1 \sqcup \cdots \sqcup \neg C_{k-1} \sqcup C_k$.

Table 6.1: PSAT performance on random propositional clauses

# atoms	# clauses	Time(s)	CG Total (s)	# columns
200	250	27.32	18.46	164.6
200	500	102.43	61.73	228.2
200	750	263.06	140.79	264.6
200	1000	396.8	185.22	274.6

growth in the number of generated columns which is the case with Hansen's technique. One possible reason for that is that we use exact optimization methods for computing each improving column while Hansen and Perron use the variable neighborhood heuristics to optimize non-linear 0-1 programs. In fact, the number of columns increases only very gently, most probably due to the fixed signature size. This will be further investigated in subsequent experiments (see especially Section 6.5.2).

At the same time the experiment reveals that the average time it takes to generate a column may increase non-polynomially. Since the number of variables in the column generation model, i.e. the MILP program (5.4), depends only on the signature size and therefore stays constant, this suggests that the program becomes harder for some other reason. We leave tuning of this program for future work.

6.3.2 Random Concept Hierarchies

The next two experiments evaluate the algorithm in the presence of TBoxes but they are restricted to propositional concept hierarchies. The general aim is to understand whether presence of classical knowledge slows the algorithm down. More specifically, our objective is to understand how well the hybrid algorithm captures propositional TBox structures and how the growing amount of classical knowledge influences the performance. Here we also switch to mostly conditional PTBox constraints which are more useful means of probabilistic modeling.

Concept Hierarchy Generation

For these experiments propositional TBoxes are generated using the following two step procedure:

1. Signature generation,

2. Generation of random TBox axioms to ensure the required subsumption density.

The first step is simply a generation of the required number of concept names to appear in conditional constraints. The second step is more interesting: it amounts

to adding a number of subsumption axioms such that the resulting TBox has the required subsumption density. The difficulty here is that the density metric counts both asserted and inferred subsumptions. Due to transitivity of the subsumption relation the number of inferred subsumptions may grow rapidly with the number of added axioms (especially for TBoxes with high density). Our density generation algorithm deals with this problem by updating the concept graph every time the new axiom is added. It provides two important guarantees: First, it terminates immediately after the required density has been reached. Second, it prevents generation of subsumption loops, i.e. sets of equivalent concepts. Such sets are undesirable since any reasoning algorithm can simply collapse them into a single concept.

It is the case that the algorithm may output a TBox which density is slightly higher than the required. However, for TBoxes with density not exceeding 25 the overshoot is not higher than 1-2 on average, so it cannot seriously skew the results. Also this problem can be rectified by generating disjointness axioms instead of subsumption (since the disjointness relation is not transitive).

Results

The first experiment evaluates the PSAT algorithm on increasingly large PTBoxes. Both classical and probabilistic parts are growing while the subsumption density of TBoxes was kept at 10 (i.e. each concept from the signature participates in roughly 10 literal subsumptions). In contrast to the experiment with random clauses 90% of PTBox constraints are conditional. The results are presented in Table 6.2.

Table 6.2: PSAT performance on random concept hierarchies of variable size and fixed subsumption density

Signature size	PTBox size	Density	Time(s)	CG Total (s)	# columns
125	250	10	50.41	45.79	95
250	500	10	117.9	95.75	178.5
325	750	10	232.34	154.19	264
500	1000	10	467.12	240.4	380
625	1250	10	711.3	318.46	470.5

Comparing these results to those obtained with collections of unconditional clauses it can be seen that neither the presence of classical knowledge nor conditional statements have substantially influenced the results. The differences in term of both total time and the column generation time are not substantial for PTBoxes with the same number of constraints in spite of larger signatures in this experiment. Larger signatures could be partly compensated by the lack of compound concept expressions in

conditional constraints, which makes the column generation program simpler.

The second experiment aims to study the influence of subsumption density on PSAT performance in *isolation*. Here the number of constraints and the signature size are kept fixed (500 and 250 respectively) while subsumption density of the TBoxes is varied between 2 and 25. Specifically, the objectives of this experiment is, first, to test whether the increasing size of the TBox leads to a substantial increase of the CG time due to extra inequalities in the MILP model and second, to check if convergence improves due to the reduced space of all possible worlds.

Table 6.3: PSAT performance on random concept hierarchies of variable subsumption density and fixed size. The column "MILP size" specifies the number of variables and inequalities in the MILP program (5.4) used to generate columns.

Sig. size	PTBox size	Density	MILP size	Time (s)	CG Total (s)	# columns
250	500	2	1200 x 2552	116.93	93.98	187.5
250	500	5	1200 x 3304	111.97	90.78	180
250	500	10	1200 x 4452	113.96	92.67	177.4
250	500	15	1200 x 5570	121.49	99.82	180.6
250	500	20	1200 x 7768	124.1	101.77	176.25
250	500	25	1200 x 8122	103.36	84.27	145

The results are presented in Table 6.3. They show that increasing subsumption density does not lead to a substantial increase in CG Total time (or even CG Total divided by the number of columns) despite the fact that the MILP model grows significantly—from 2552 to 8122 inequalities—because there is more TBox structure to be captured. This further increases the confidence that our PSAT algorithm is robust and largely scalable with respect to the amount of classical knowledge. The results are, however, inconclusive regarding convergence. While it does seem to be better for instances with high density, i.e., 25, the difference is not substantial. This may indicate that better stabilization strategies could be required.

6.4 Random Bayesian Knowledge Bases

In this section we present the evaluation of the PSAT algorithm on knowledge bases that have been produced from randomly generated Bayesian networks of variable shape. While we do not explicitly aim at competing with Bayesian inference methods P-$\mathcal{SROIQ}$, as a formalism, could be interesting in the context of reasoning with Bayesian models. More specifically, P-$\mathcal{SROIQ}$ provides excellent means for integrating probabilistic knowledge with OWL ontologies which can be explored in domains where both ontologies and Bayesian models have been developed (one such example is the ontology-based Bayesian network approach to clinical practice guidelines [190]). In particular,

P-$\mathcal{SROIQ}$ can be used for:

- Finding contradictions between classical background knowledge captured in ontologies and probabilistic knowledge represented in Bayesian networks, or proving consistency. For example, if the ontology entails that symptom `S` is always present in patient with disease `D` than $P(D|C) < 1$ should not follow from the Bayesian network (at least such contradictions should be automatically discoverable).
- Comparison of multiple Bayesian networks in presence of an ontology.
- Construction of Bayesian networks from initially developed probabilistic ontologies, for instance, for the sake of performance (see [13, 74]).

A formal theory of using P-$\mathcal{SROIQ}$ and its extensions for performing these tasks is beyond the scope of this thesis. However, it seems clear that to be practical such approach requires adequate tool support, in particular, a sufficiently scalable reasoner. P-$\mathcal{SROIQ}$, as it stands, does not provide enough representational capabilities for faithful handling of Bayesian networks as logical knowledge bases. Specifically, conditional independence assertions, which lie in the basis of Bayesian networks, are not representable using conditional constraints. However, as shown in [5], Nilsson-style probabilistic logics can serve as bases for the more expressive *Bayesian logic*. Furthermore, column generation methods, which are central to our PSAT algorithm, can still be used for solving PSAT in the Bayesian Logic although they need to be complemented by other techniques, such as Benders decomposition. From this perspective, even though scalability of P-$\mathcal{SROIQ}$ reasoners is not a sufficient condition of practicality of Bayesian logic, it could well be a solid first step towards that goal.

In addition to evaluating scalability, the experiments aim to test whether the PSAT algorithm is sensitive to *tree width* which is known to be the key parameter for many Bayesian inference algorithms, for example, belief propagation [163]. Tree width is one less than the size of the largest clique over all possible triangulations of the graph representing the undirected version of the Bayesian network. Informally, it characterizes how well the graph can be decomposed onto a tree [171]. Most of the exact and approximate algorithms are worst case exponential in tree width thus practical networks are often constructed to limit connectivity between nodes.

Since the PSAT algorithm is not based on any sort of message passing, we hypothesize that it should not be directly sensitive to tree width. This is an interesting property which, if supported by the experiments, suggests that future modelers need not be worried about the "shape" of their knowledge bases (if visualized as graphs). Thus it could potentially enable modeling patterns which could seem dangerous from a Bayesian network perspective.

6.4.1 Knowledge Base Generation

For this experiment we produce P-$\mathcal{SROIQ}$ knowledge bases using a two phase process: First, random Bayesian networks with varying tree width are generated. Second, the generated networks are approximately translated into P-$\mathcal{SROIQ}$.

Recall that Bayesian network is a directed acyclic graph (DAG) with N nodes where each node X is a discrete random variable with a finite set of values. For each node X the network must specify a full conditional probability $p(X|P_X)$ where P_X is the set of all parents of x. The generation of a probability distribution is relatively straightforward (due to the locality property) therefore the main problem is to generate *uniformly* distributed random DAGs.

We use the algorithms developed by Ide and Cozman [91] for performing this task. First, the algorithms give reasonable guarantees that the structure of Bayesian networks has been *uniformly* sampled from the large space of all DAGs with specified number of nodes and values per node. Second, the algorithms allow us to control the tree width parameter during generation. This is done by checking tree width as the DAG is being generated and, consequently, a heuristic approach is used because determining the tree width is an NP-complete problem. However, the authors observed that the induced width heuristics approximates tree width very well. In addition, we try to minimize any unwanted effects by generating multiple networks with the same tree width. Finally, the algorithms are highly configurable and an implementation is available.[7]

In this experiment all random variables in Bayesian networks have values *true* and *false* and are treated as concept names. Thus a conditional probability, e.g. $P(X = true|X' = false, X'' = true) = p$, is interpreted as a statement "the probability that a random object is an instance of X given that it is an instance of X'' and not an instance of X' is p". Recall from Section 2.3.1 that it corresponds to the semantics of conditional constraints in P-$\mathcal{SROIQ}$.

Once the network has been generated, it is transformed into a P-$\mathcal{SROIQ}$ knowledge base in the following straightforward way: First, all variable names N are translated into concept names. Second, each conditional probability $P(X_0 = x_0|\{X_i = x_i\}) = p$, where $x_i \in \{true, false\}$, is represented as a conditional constraint $(Y_0|\bigsqcap Y_i)$ where Y_i is equal to X_i if $x_i = true$ and $\neg X_i$ if $x_i = false$.

This translation is approximate because it ignores conditional independencies encoded in the structure of the networks. As mentioned above, representation of such assertion requires non-linear statements, for example, $(Y_0|Y_1 \sqcap Y_2) = (Y_0|Y_1)$. Even though P-$\mathcal{SROIQ}$ does not allow us to fully capture the semantics of the network, it still supports useful patterns of approximate reasoning about the network. For instance, if the approximate translation of the network happens to contradict an ontology then

[7] http://www.pmr.poli.usp.br/ltd/Software/BNGenerator/

the exact translation will also contradict the same ontology (the converse is obviously false).

In contrast to the other experiments, we are less interested in measuring PSAT performance as a function of knowledge base size. Instead, we generate Bayesian networks in such a way that the resulting knowledge bases are close to the scalability limit of the PSAT algorithm, i.e. around 1000 conditional constraints. One reason is that we are more interested in varying tree width which has obvious impact on the number of conditional constraints, and it is not always possible to reliably vary tree width while keeping the number of constraints small.

We generated 10 random Bayesian networks for each value of tree width: 5, 10, 15, and 20. The number of nodes in the network, which corresponds to the size of probabilistic signature of the resulting knowledge base, was kept at 100. The maximum node degree was fixed at 5 while the limit on the total number of edges was varied between 180 and 250 to ensure that the total number of constraints is around 1000. As a result, more than 95% of randomly generated networks were translated into P-$\mathcal{SROIQ}$ KBs with the number of constraints between 900 and 1100. The remaining 5% were discarded and regenerated.

6.4.2 Results

The evaluation results are presented in Table 6.4.

Table 6.4: PSAT performance on translations of Bayesian networks with variable tree width into P-$\mathcal{SROIQ}$.

Tree width	Time (s)	CG Total (s)	# columns
5	1441.8	765.6	727.5
10	2062.1	1186.9	836.2
15	1930.7	1093.1	762.8
20	1660.8	985.2	782.5

The main outcome is that the current implementation is scalable enough to handle approximate representations of Bayesian networks of about 100 nodes. The algorithm exhibits pretty good convergence despite the fact that no TBox exists. This is likely to be due to the relatively small probabilistic signature of 100 concepts. The average time to generate a column (i.e., CG Total divided by the number of generated columns) is higher than in the previous experiments, which might appear surprising given lack of any TBox structure. It can be explained by the fact that evidence concepts in conditional constraints are usually conjunctive expressions (conjunctions of parent nodes) which are abbreviated by automatically generated concept names. Definitions of those

autogenerated concepts give rise to large artificial TBoxes which, in turn, increases the number of constraints in the column generation program.[8]

The results support our initial expectation that the algorithm should not be sensitive to tree width. Given that signature size and the total number of constraints are fixed, tree width does not seem to have any impact on performance or scalability. Consequently, when considering the feasibility of using P-$\mathcal{SROIQ}$ for approximate reasoning about Bayesian networks, or other graphical probabilistic models, one can only pay attention to the size and the number of local conditional probability distribution and ignore such considerations as shape or density of the graph. More broadly, we suggest that analyzing graph-like structure of P-$\mathcal{SROIQ}$ knowledge bases is not a generally reasonable way of estimating their practical complexity for PSAT/TLogEnt algorithms based on column generation.

6.5 Probabilistic Extensions of Real Ontologies

This section describes the evaluation of the PSAT algorithm on probabilistic knowledge bases generated on top of real-life ontologies. Since P-$\mathcal{SROIQ}$ was designed as an extension of the DLs behind OWL and compatibility with OWL is declared as one of its major advantages, it is critical to ensure that any reasoning algorithms can successfully deal with probabilistic extensions of real OWL ontologies, not just randomly generated clauses or propositional taxonomies. In that sense these experiments are central for proving the practicality of our algorithms, in particular, the PSAT decision procedure.

More precisely, the experiments described in this section have been designed to pursue the following objectives:

- Evaluate the scalability the PSAT algorithm on probabilistic extensions of real ontologies.
- Evaluate the robustness of the algorithm and understand its sensitivity to the main characteristics of the input KBs. We are mostly interested in finding out whether such characteristics of real ontologies as size, representation language, subsumption density, etc. may cause substantial performance variability.
- Evaluate the effectiveness of the major optimization strategies on probabilistic extensions of real ontologies. Some optimizations which work well on propositional KBs may be less effective or run into troubles on KBs with rich classical part. For example, finding all minimal unsatisfiable subexpressions of a conjunctive concept expression may get substantially more difficult in presence of a large

[8] TBoxes may lead to some column candidates being invalid but, in this case, Pronto can avoid it due to the propositional absorption optimization, but again, at the expense of larger MILP instances.

and rich (i.e., with numerous non-trivial entailments) TBox. The same holds for exploiting concept hierarchy techniques.

- Compare the performance and scalability of the algorithm on real ontologies to the same metrics computed for random propositional ontologies.

Differently from the previous experiments, in which all KBs were satisfiable, here we also present experiments with unsatisfiable KBs. This is done primarily to evaluate the effectiveness of our early conflict detection method described in Section 5.1.5.

6.5.1 Ontology Selection

The first important step of this evaluation is selection of appropriate real ontologies. A wide range of ontologies are currently available, even if attention is restricted to those represented in one of OWL syntaxes, thus establishing some selection criteria is necessary. While we do not aim at a comprehensive study of OWL ontology landscape with regard to complexity of probabilistic reasoning, we used the following guidelines when picking the ontologies:

- Some of the ontologies should be represented in an expressive language. One of our long-term goals is to provide tools for reasoning over probabilistic extensions of OWL ontologies, so we wanted to evaluate the performance on ontologies which make use of as many features of OWL 2 as possible. At the same time ontologies represented in a lightweight fragment of OWL 2, such as OWL EL, also need to be included, especially given the fact that they are getting increasingly widespread in some important domains, such as medical informatics.

- Some of the ontologies should have reasonably large TBoxes with at least few hundred concept subsumption, equivalence or disjointness axioms and some object roles. By "non-trivial" we mean that the TBox should have entailments that cannot be discovered simply by traversing the concept hierarchy. This is essential for evaluating the effectiveness of our iterative approach to column generation according to which TBox structure is captured in a lazy fashion by interaction with a $\mathcal{SROIQ}$ reasoner.

- The ontologies should also have at least 500 concepts in the TBox to show that the reasoner can handle large probabilistic signatures.

- Finally, the ontologies should allow for reasonably efficient reasoning using currently available OWL reasoners. It is clear that any evaluation of a PSAT algorithm is futile if the classical part of the ontology is too hard for the OWL reasoner (since the PSAT problem involves a series of SAT tests in the target DL).

This requirement rules out some too big ontologies, such as Galen or SNOMED CT, or some very hard ontologies, such as the Family ontology.

- Ideally, the ontologies should be "in service", i.e. have been created for and be in use by real applications as opposed to educational or experimental purposes. In that case they are more likely to encompass common and useful modeling patterns.

Using these guidelines we selected the following six ontologies from different domains:

The NCI Anatomy Ontology (NCI) is a part of the NCI Thesaurus which describes human anatomy. It is relatively simple (represented in $\mathcal{ALE}+$) but large as it contains more than 3300 concepts and 5423 concept inclusion axioms. We use the same version of the ontology which was been used in the Ontology Alignment Evaluation competition in 2009.[9]

The Subcellular Anatomy Ontology (SAO) is the ontology from the neuroinformatics domain describing cellular and subcellular structures, supracellular domains, and macromolecules.[10] It contains 737 concepts, 915 subsumption, 4 equivalence, and 1580 concept disjointness axioms, 36 object properties and 47 data properties. It is represented in $\mathcal{SHIN}(D)$.

The Process Ontology is a part of the SWEET (Semantic Web for Earth and Environmental Terminology) collection of ontologies developed by NASA to provide semantic support for various Earth science projects.[11] It contains 1537 concepts, 1922 subsumption, 84 equivalence, and 1 concept disjointness axioms, 102 object properties and 19 data properties. It is represented in $\mathcal{ALCHOF}(D)$.

The Sequence Ontology with Composite Terms (SO-XP) defines terms and relationships used to describe features and attributes of biological sequence as well as cross-product definitions for composite terms.[12] It is a deliverable of the Gene Ontology Project and the Open Biomedical Ontologies (OBO) experiment. It contains 1660 concepts, 1709 subsumption, 198 equivalence, and 21 concept disjointness axioms and 22 object properties. Its representation language is $\mathcal{SHI}$.

The Teleost Anatomy Ontology (TAO) is a multi-species anatomy ontology for teleost fishes.[13] It contains 2229 concepts, 3 object properties and 3406 concept subsumption axioms. It is represented in $\mathcal{EL}+$ (the OWL 2 EL profile).

[9] `http://oaei.ontologymatching.org/2009/results/anatomy/`
[10] `http://ccdb.ucsd.edu/CCDBWebSite/sao.html`
[11] `http://sweet.jpl.nasa.gov/ontology/`
[12] `http://wiki.geneontology.org/index.php/SO:Composite_Terms`
[13] `https://www.nescent.org/phenoscape/Teleost_Anatomy_Ontology`

The Cell Type ontology (CO) is a structured controlled vocabulary for cell types constructed for model organism and other Bioinformatics databases.[14] It contains 857 concepts, 1 object property and 1263 concept subsumption axioms. It also falls into the OWL 2 EL profile.

None of these ontologies is propositional or small and simple enough to consider their propositionalization and a subsequent use of a propositional probabilistic SAT solver as a feasible alternative. None of the previously developed PSAT algorithms is capable of dealing with thousands of classical axioms in addition to a comparable number of probabilistic formulas.

6.5.2 Results

For each ontology selected to serve as the classical part of the PTBox we have generated the probabilistic part by selecting random probabilistic signature and generating random probability intervals. We have performed two series of experiments aimed at testing the scalability, robustness, and correctness of the PSAT algorithm.

Series I: Satisfiable PTBoxes

The first series of experiments was run on satisfiable PTBoxes to evaluate scalability, robustness, and also investigate whether the results vary substantially across the ontologies. The series includes four experiments, three of which measure performance metrics as functions of such parameters as the size of probabilistic signature and the number of PTBox constraints, while the fourth—as functions of the proportion of unconditional constraints in the PTBox.

For each PSAT instance the probabilistic part of the PTBox was generated following the generic procedure outlined in Section 6.1.2. First, a random signature of the required size was selected. Second, the required number of random concept pairs were selected from the signature. Finally, probability intervals were generated for the concept pairs by invoking Algorithm 10.

Fixed Signature, Variable PTBox Size In the first experiment we measure performance of the PSAT algorithm as a function of PTBox size. We keep probabilistic signature size fixed at 250 concept names while varying the number of constraints in the PTBox between 250 and 1000. The proportion of unconditional constraints was fixed at 10%. The results are presented in Table 6.5.

[14] `http://obolibrary.org/cgi-bin/detail.cgi?id=cell`

Table 6.5: PSAT times for PTBoxes with probabilistic signatures of 250 concept names and variable size

Ontology	Language	TBox size	PTBox size	Density	Total time (s)	CG Total (s)	# columns
NCI	$\mathcal{ALE}+$	5423	250	1.3	151.3	22.6	99.0
			500	1.2	240.1	95.1	190.6
			750	1.3	314.9	125.7	241.8
			1000	1.2	440.4	163.4	306.0
SAO	$\mathcal{SHIN}$	2499	250	33.3	77.9	65.3	123.2
			500	33.7	160.5	134.2	246.2
			750	34.0	321.2	242.8	404.2
			1000	33.1	525.5	344.5	526.0
Process	$\mathcal{SHOF}$	2007	250	3.5	46.7	28.6	97.2
			500	3.3	124.7	91.2	180.4
			750	3.0	211.8	132.4	248.6
			1000	4.0	337.5	166.7	300.4
SO-XP	$\mathcal{SHI}$	1928	250	70.2	129.2	89.3	130.4
			500	73.5	206.1	151.7	198.2
			750	73.4	319.5	227.0	251.8
			1000	72.2	525.0	350.7	318.4
TAO	$\mathcal{EL}+$	3406	250	1.4	43.6	22.4	95.6
			500	1.2	127.4	90.2	182.4
			750	1.3	205.4	124.1	240.8
			1000	1.3	326.2	164.4	310.2
Cell Type	$\mathcal{EL}+$	1263	250	3.8	66.0	34.3	98.0
			500	3.3	138.1	91.3	182.2
			750	3.1	219.6	126.7	244.2
			1000	3.6	336.5	162.5	300.2

Variable Signature, Fixed PTBox Size In the second experiment we evaluate performance of the PSAT algorithm on PTBoxes of a fixed size but with a variable number of concepts in the probabilistic signature. The aim here is to evaluate performance as well as sensitivity of the algorithm to the signature size. PTBox size was kept fixed at 500 constraints while the signature size was varied from 100 to 500. The proportion of unconditional constraints was fixed at 10% as above. The results are presented in Table 6.6.

Variable Signature, Variable PTBox Size In the third experiment we vary both PTBox size and the size of probabilistic signature. The aim of this experiment is to evaluate performance and robustness of the PSAT algorithm in more realistic settings where larger PTBoxes typically involve larger signatures. The number of constraints was varied between 250 and 1000 as in the first experiment but signature's size was kept at 50% of the PTBox size. The number of uncoditional statements was fixed at 10% as before. The results are presented in Table 6.7.

Table 6.6: PSAT times for PTBoxes with 500 probabilistic statements and variable signature size

Ontology	Language	TBox size	Sig. size	Density	Total time (s)	CG Total (s)	# columns
NCI	$\mathcal{ALE}+$	5423	100	1.2	139.6	71.8	145.4
			200	1.2	201.7	82.7	169.6
			300	1.5	258.9	89.2	202.6
			400	1.4	305.0	84.9	224.0
			500	1.6	365.2	93.4	249.2
SAO	$\mathcal{SHIN}$	2499	100	13.2	179.3	153.1	306.0
			200	28.1	158.6	133.4	246.0
			300	41.1	165.5	138.0	232.6
			400	52.3	252.4	221.9	212.0
			500	66.0	415.2	382.2	197.0
Process	$\mathcal{SHOF}$	2007	100	2.6	95.7	73.5	146.6
			200	2.3	119.2	89.6	175.6
			300	4.0	134.3	97.4	188.8
			400	5.0	158.0	111.5	211.0
			500	6.4	180.3	124.0	234.8
SO-XP	$\mathcal{SHI}$	1928	100	27.1	118.7	89.4	152.0
			200	59.6	178.3	132.6	186.4
			300	86.5	266.7	203.3	208.4
			400	113.8	330.6	250.8	224.6
			500	139.4	468.6	370.5	252.8
TAO	$\mathcal{EL}+$	3406	100	1.2	91.9	68.9	139.8
			200	1.2	107.8	76.6	156.2
			300	1.2	124.9	84.0	193.6
			400	1.7	139.7	88.6	224.4
			500	1.7	160.2	95.9	258.0
Cell Type	$\mathcal{EL}+$	1263	100	1.5	96.5	69.0	141.6
			200	3.0	126.6	86.0	172.4
			300	3.7	155.5	101.1	198.4
			400	5.8	170.6	105.8	200.4
			500	6.3	218.8	139.3	252.6

Variable Proportion of Unconditional Constraints The final experiment of this section evaluates robustness of the PSAT algorithm with respect to the relative number of unconditional constraints in the PTBox. Both signature size and PTBox size were fixed at 250 and 500 respectively. The proportion of unconditional constraints was varied between 10% and 50%. The results are presented in Table 6.8.

Summary The first, and the major, conclusion that can be made from the evaluation results is that the algorithm is *robust*, i.e. it behaves quite predictively on satisfiable PTBoxes with varying parameters. No combination of the main parameters causes it to hit the worst case. It robustly scales to 1000 probabilistic statements defined over 500 concepts from expressive real ontologies.

The second observation is that PTBoxes built over the SAO and the SO-XP ontologies tend to be harder for the algorithm than the rest. The difference is especially

Table 6.7: PSAT times for PTBoxes with varying number of statements and signature size

Ontology	Language	TBox size	PTBox size	Density	Total time (s)	CG Total (s)	# columns
NCI	$\mathcal{ALE}+$	5423	250	1.1	100.2	32.8	83.4
			500	1.3	239.5	93.9	186.4
			750	1.5	429.1	157.4	301.4
			1000	2.0	745.1	231.6	418.4
SAO	$\mathcal{SHIN}$	2499	250	15.8	77.1	68.4	129.4
			500	32.4	178.2	149.3	276.4
			750	47.7	375.3	300.0	341.2
			1000	63.6	1360.2	1176.1	425.4
Process	$\mathcal{SHOF}$	2007	250	1.9	51.0	39.4	88.6
			500	2.6	119.9	87.0	176.4
			750	4.4	240.9	144.4	275.2
			1000	5.2	479.7	236.4	404.8
SO-XP	$\mathcal{SHI}$	1928	250	33.1	61.3	40.3	76.0
			500	71.7	197.1	144.2	189.0
			750	107.7	449.5	323.3	307.6
			1000	138.5	921.6	644.3	423.4
TAO	$\mathcal{EL}+$	3406	250	1.2	50.2	37.1	89.4
			500	1.2	125.8	89.4	179.8
			750	1.5	252.5	149.5	287.8
			1000	1.9	544.7	238.1	431.8
Cell Type	$\mathcal{EL}+$	1263	250	2.5	57.2	39.2	89.2
			500	3.7	137.9	91.7	182.6
			750	4.4	283.5	158.9	296.4
			1000	5.5	487.7	220.3	384.2

visible in Table 6.6 and Table 6.7, i.e. where signature is varied. While the total number of generated columns is approximately the same, it is substantially harder to generate a column when signature size is over approximately 200 concepts (for the SO-XP ontology) and 300 concepts (for the SAO ontology). This is illustrated by the CG Total times, which rise sharply for those ontologies.

The explanation for this "thrashing" effect is the high richness of those two ontologies as revealed by the subsumption metric. High density values mean that concepts, which are randomly selected from the TBox, are often engaged in subsumption or disjointness relationships. These relationship need to be captured in the MILP model. However, if the number of the relationships is high not all corresponding linear inequalities will be created when exploiting the concept hierarchy, as explained in Section 5.1.5, in order to prevent memory exhaustion.[15] Consequently, invalid column candidates are more likely to be generated inside Algorithm 4. Each appearance of an invalid column

[15]Given the available RAM (2GB) we set the limit of the height of the MILP model to 15,000 inequalities. This is sufficient to capture all subsumptions following from the classified TBox for the Process, T-A and Cell Type ontologies, but not for the SAO or SO-XP ontologies. The problem is especially visible for the SO-XP ontology for which Algorithm 4 computed more than 100 invalid columns for an average PTBox of 500 constraints and 500 concepts in the signature.

Table 6.8: PSAT evaluation results on PTBoxes with fixed size and variable number of unconditional constraints. U% stands for the proportion of unconditional constraints in the PTBox.

Ontology	Language	TBox size	U%	Density	Total time (s)	CG Total (s)	# columns
NCI	$\mathcal{ALE}+$	5423	10	1.3	239.5	93.9	186.4
			20	1.2	240.3	93.1	194.5
			30	1.2	248.6	97.1	212.0
			40	1.2	259.2	101.5	258.5
			50	1.3	249.4	87.0	255.0
SAO	$\mathcal{SHIN}$	2499	10	32.4	178.2	149.3	276.4
			20	33.1	191.4	162.6	299.5
			30	31.4	265.0	229.1	411.5
			40	25.1	263.3	224.9	404.5
			50	33.4	309.9	266.5	434.5
Process	$\mathcal{SHOF}$	2007	10	2.6	119.9	87.0	176.4
			20	2.3	136.4	99.5	194.0
			30	2.8	152.1	112.1	223.5
			40	2.4	144.4	102.9	220.0
			50	2.6	142.2	96.2	230.5
SO-XP	$\mathcal{SHI}$	1928	10	71.7	197.1	144.2	189.0
			20	73.2	268.7	209.2	240.0
			30	70.2	251.9	193.2	253.0
			40	71.1	323.9	259.0	306.0
			50	70.9	305.7	237.7	312.5
TAO	$\mathcal{EL}+$	3406	10	1.2	125.8	89.4	179.8
			20	1.2	129.6	90.6	187.0
			30	1.3	131.2	90.9	185.5
			40	1.1	144.3	99.3	222.0
			50	1.5	154.9	98.7	264.5
Cell Type	$\mathcal{EL}+$	1263	10	3.7	137.9	91.7	182.6
			20	3.4	166.4	114.9	224.5
			30	3.3	164.2	112.2	218.5
			40	3.6	158.4	104.9	215.0
			50	3.3	165.8	107.5	243.0

will trigger the computationally intensive process of computing the unsatisfiability core to find all (or some) minimal unsatisfiable expressions. This is the main reason why thrashing occurs. Such effect can be mitigated by either increasing the amount of available memory or employing a more intelligent approach to compute the initial set of inequalities for the MILP model.

The third outcome is that the number of columns generated by Algorithm 4 does not seem to grow exponentially with either size of the PTBox or size of the probabilistic signature. This suggests that the PSAT algorithm may well scale beyond 1000 conditional constraints. We have not yet extended the experiments beyond 1000 for two reasons. First, it is time consuming to generate *satisfiable* probabilistic KBs of that size over complex ontologies because it requires computing a high number of possible worlds. Second, it is currently unclear what the real requirements for scalability of

P-$\mathcal{SROIQ}$ reasoners are since the only handcrafted P-$\mathcal{SROIQ}$ ontology (the BCRA ontology) is well below that limit.[16]

Another observation is that convergence of the algorithm does not seem to depend on TBox richness. This is slightly surprising because richer TBoxes tend to reduce the total space of valid columns, so some benefits for column generation were anticipated. Our conjecture is that the outcome could be due to first, the approximative nature of the subsumption density metric, and second, the stabilization technique used to improve convergence. Although we have not conducted a thorough experimentation with different stabilization techniques, our experience is that the iterative stabilization, similar to the one used by Hansen and Perron [81], is significantly less efficient on PTBoxes with low density. However the effect does not seem to hold for the straight stabilization that has been used for all our experiments.

Finally, it is interesting that while the algorithm is also robust with respect to the number of unconditional constraints, its convergence slightly worsens as the proportion increases. A likely explanation of this phenomenon is that unconditional constraints place "stronger" restrictions on probabilistic models than conditional ones because they cannot be satisfied vacuously, i.e., by assigning zero probability to the evidence concept (since it is $\top$). This apparently causes the algorithm spend more iterations to find a "non-trivial" probabilistic interpretation.

Series II: Unsatisfiable PTBoxes

Finally, we present the PSAT evaluation results on unsatisfiable PTBoxes to understand the effectiveness of the algorithm, in particular, the early conflict detection (ECD) method described in Section 5.1.5, in the presence of inconsistent knowledge. Good performance on such PTBoxes is critically important in non-monotonic reasoning, in particular, for constructing conflict graphs, z-partition and computing lexicographic entailment. However, in this experiment we are only interested in measuring the effectiveness of *detecting* conflicts as opposed to discovering or extracting them from the knowledge base. Evaluation of the diagnosis algorithm on unsatisfiable fragments of CADIAG-2 knowledge base is presented in Section 7.2.2.

The structures of random PTBox used in this experiment are exactly the same as for those described in the "Variable Signature, Variable PTBox Size" paragraph above. We vary two parameters: the number of constraints in the PTBox and the size of probabilistic signature. The ratio between the signature size and the PTBox size was fixed at 0.5. The proportion of unconditional constraints was kept fixed at 10%. Probability intervals were generate randomly as described at the end of Section 6.1.2.

[16]The probabilistic formalization of CADIAG-2 KB is larger but it has not been designed as a probabilistic ontology in P-$\mathcal{SROIQ}$ or even as a probabilistic knowledge base in general.

10% of unconditional statements is enough to ensure that most of PTBoxes generated in this way are unsatisfiable (those which by accident turned out to be satisfiable have been excluded from the experiment).

In addition to the standard performance measures such as time and the number of generated columns, here we also measure the proportion of PTBoxes whose unsatisfiability has been detected by ECD and the average number of times ECD has been triggered but failed to detect a conflict (so called *false runs*). As explained in Section 5.1.5, each ECD invocation incurs an overhead because it involves solving the PSAT problem for a subset of the original PTBox. Even though we limit the size of such subsets in order to reduce the negative impact of false runs, it is still necessary to account for them when measuring the efficiency of the technique.

The results are presented in Table 6.9. As in previous experiments 10 PSAT instances have been solved for each size. The following observations can easily be made:

Table 6.9: PSAT times on unsatisfiable PTBoxes. H is the proportion of problem instances for which unsatisfiability has been detected by ECD. F is the average number of false invocations of ECD per problem instance. R is the average time savings (in %) that are due to ECD.

Ontology	Sig. size	PTBox size	Density	H	F	R (%)	Total time (s)	CG Total (s)	# columns	Time (ECD Off)
NCI	125	250	1.1	0.0	2.3	-8.3	97.4	23.4	63.5	89.9
	250	500	1.3	0.0	0.3	-1.3	212.1	63.2	124.5	209.4
	375	750	1.5	0.0	0.5	-3.5	351.6	85.5	164.5	339.8
	500	1000	2.0	0.2	1.8	-7.0	661.7	160.4	275.0	618.5
SAO	125	250	15.8	0.8	0.4	6.5	24.1	16.4	31.8	25.8
	250	500	32.4	1.0	0.4	6.8	48.8	31.0	55.5	52.3
	375	750	47.7	1.0	0.2	17.6	84.4	54.6	53.5	102.4
	500	1000	63.6	1.0	1.4	31.5	175.2	128.6	44.9	255.8
Process	125	250	1.9	0.0	3.2	-16.7	25.4	11.9	35.0	21.7
	250	500	2.6	0.8	0.6	26.5	32.5	14.3	33.1	44.3
	375	750	4.4	0.8	0.6	25.6	69.1	30.3	61.0	92.8
	500	1000	5.2	0.8	1.2	37.2	94.5	37.5	67.5	150.6
SO-XP	125	250	33.1	0.2	0.2	7.4	24.5	5.3	11.6	26.5
	250	500	71.7	1.0	0.6	21.8	62.7	27.6	42.2	80.2
	375	750	107.7	1.0	0.6	43.6	104.4	58.4	54.2	185.1
	500	1000	138.5	1.0	0.4	56.5	200.7	141.9	49.1	461.8
TAO	125	250	1.2	0.0	5.0	-14.9	58.3	40.9	100.5	50.7
	250	500	1.2	0.4	2.0	-1.4	114.1	68.3	136.8	112.6
	375	750	1.5	0.6	2.6	12.6	131.3	61.1	123.7	150.2
	500	1000	1.9	0.8	1.6	28.2	203.7	77.5	145.1	283.6
Cell Type	125	250	2.5	0.0	4.0	-8.6	53.7	32.8	76.5	49.5
	250	500	3.7	0.6	0.2	9.5	64.4	28.5	60.6	71.2
	375	750	4.4	0.8	0.4	24.7	113.5	50.8	97.9	150.7
	500	1000	5.5	0.8	1.2	31.1	173.7	70.3	121.2	252.1

First, for most of the ontologies it takes substantially fewer (often by an order of magnitude) columns to prove unsatisfiability than to prove satisfiability. One may argue that this is a non-surprising outcome because it is sufficient to stumble upon

a single conflict to terminate the process. However, the PSAT algorithm does not work by inspecting subsets of the KB and trying to detect a conflict, especially if the ECD optimization is off. The algorithm only terminates if i) the current optimal value of the main LP program (5.5) is less than 1, and ii) an improving column cannot be generated. So the algorithm can hit its worst case in at least two scenarios. First, it may generate an exponential number of only marginally improving columns. It may bring the optimal value of (5.5) arbitrarily close to 1 but still never reach it. Second, even if at some step no improving column exists, it may take Algorithm 4 an exponential number of steps to prove that. For instance, Algorithm 4 can generate an exponential number of invalid column candidates before concluding that no *valid and improving* column exist. The result that the algorithm avoids both these traps is not trivial although it can be observed that column generation process is more computationally intensive for unsatisfiable PTBoxes (see the "CG Total (s)" column in Table 6.9).

The second observation is that in contrast to satisfiable PTBoxes convergence does not seem to depend on the number of constraints. For most ontologies the number of generated columns is approximately the same regardless of PTBox size. A possible explanation of this result is that the total number of conflicts is most probably higher for larger PTBoxes, so it may well be easier for the algorithm to stop at some of them. At the same time it does not mean that the number of constraints is not an important parameter. It still strongly influences the time it takes to generate a column and, consequently, the total time.

Third, it is interesting that the ECD technique appears more effective for larger PTBoxes and PTBoxes over richer ontologies, namely, SAO and SO-XP. For the "weak" ontologies (Process, TAO and Cell) and PTBoxes of 250 constraints it always fail to detect a conflict while performing 3–5 false runs per PSAT instance. This leads to a negative, albeit small, impact of the technique as can be seen from the R(%) column. This never happens for PTBoxes of 750 or more constraints or rich ontologies. It is likely that the effectiveness depends on the total number of conflicts which, as mentioned in the previous paragraph, is higher for larger PTBoxes. This can also explain dependence on richness because TBox structure can also induce conflicts in random knowledge bases by ruling out more worlds.

Finally, it appears that the technique is not effective for the NCI ontology and the difference in the number of generated columns between unsatisfiable and satisfiable (see Table 6.7) PTBoxes is insignificant. Our general observation is that the column generation process for NCI does not "stall" as often happens for other unsatisfiable PTBoxes, so ECD simply is not triggered thus causing no improvement and very little harm. One possible reason is that NCI TBox has the simplest structure among all ontologies. Further investigation is left for future research.

6.6 Summary of Results

In this chapter we have presented the results of an extensive performance evaluation involving two key reasoning services (PSAT and TLexEnt) and a wide range of various classes of probabilistic knowledge bases. The following major conclusions can be made:

Scalability with respect to probabilistic knowledge The PSAT algorithm demonstrated the capability of handling hard, i.e. non-trivially satisfiable, PTBoxes of 1000 probabilistic statements in all experiments. Moreover, in most of the experiments the algorithm does not exhibit a clearly exponential trend. The algorithm performs comparably to the previously developed propositional PSAT solvers on propositional PSAT instances, however, it is a lot more generic with respect to the expressivity of classical (non-probabilistic) knowledge.

Scalability with respect to classical knowledge It is worth stressing that the algorithm does not exhibit considerable sensitivity with respect to the amount of classical knowledge (provided the classical $\mathcal{SROIQ}$ reasoner can handle SAT). The only issue is the number of inequalities for (5.4) which *may* be required to fully capture the TBox structure. However, the experience suggests that the algorithm typically terminates before computing an excessive (for the given amount of available memory) number of inequalities.

Robustness and Predictability The PTBoxes used in different experiments vary widely in their characteristics, from the number of constraints and the size of probabilistic signature to expressivity and richness of the classical part. While some PTBoxes appear to be harder than others (the prime example are the ontology alignments while the easiest are the PTBoxes over random concept hierarchies) the difference is less than a single order of magnitude. Furthermore, the algorithm's performance is quite predictable for each class of PTBoxes as the standard deviation for both total time and the number of generated columns does not exceed 10% (one exception is the experiment with unsatisfiable PTBoxes for which the ECD heuristics sometimes detects a conflict after having generated only about 10 columns).

Applicability The algorithm demonstrated its applicability in use cases such as validating probabilistic ontology alignments and reasoning about Bayesian networks. Another application of the PSAT and diagnosis algorithms to naturally occurring knowledge bases, namely the CADIAG-2 KB, is described in Section 3.2. While these experiments by themselves are not sufficient to claim practicality, they should encourage modelers to attempt to model other problems using P-$\mathcal{SROIQ}$.

Optimizations Effectiveness All optimization techniques described in Section 5.1.5 play major roles in the experiments. Vast majority of inequalities for the column generation model are pre-computed by the *Exploiting Concept Hierarchy* and *Propositional Absorption* techniques which prevent Algorithm 4 from computing numerous invalid columns. The *Optimistic Inequality Generation* optimization is indispensable for dealing with hard TBoxes for which computing all unsatisfiable subexpressions of a conjunctive concept expression is highly intractable (mostly due to a high number of those). In particular, the PSAT algorithm fails to terminate within 30 minutes on PTBoxes with 1000 constraints over the SAO and the SO-XP ontologies if this technique is off. The *Stabilization* technique is the key to an acceptable convergence. In the absence of stabilization the algorithm generates more than 5,000 columns for PTBoxes with weak classical part, for example, for sets of unconditional clauses. Finally, the *Early Conflict Detection* method significantly improves the performance on unsatisfiable PTBoxes (up to 50% on large instances) while not causing any substantial harm on satisfiable PTBoxes. Although it does not push forward the scalability limits it is very important during non-monotonic, such as lexicographic, reasoning based on resolving contextual conflicts in background probabilistic knowledge.

Memory Consumption The PSAT algorithm is memory intensive due to the following two reasons. First, it makes heavy use of the $\mathcal{SROIQ}$ reasoning algorithms which have exponential memory requirements. Second, it may require a non-polynomial number of linear inequalities to capture TBox structure in the MILP model (5.4). In the experiments with some of the real ontologies (especially SO-XP) the algorithm was able to exhaust 2GB of available RAM on our machine so we had to limit the initial size of the MILP model to 15,000 inequalities. Currently this threshold is set manually but can be determined by the reasoner based on the amount of available memory.

Modularity Need Finally, the experiments helped to reveal major obstacles to further improvements in scalability. The key obstacle is the size and hardness of the MILP model (5.4) used to generate columns for the PSAT algorithm. CG Total divided by the number of generated columns is the only performance measure that appears to grow non-polynomially with the size of probabilistic KBs. According to our experiments, approximate approaches to solving (5.4) do not promise substantial improvements because the time it takes to obtain *some*, not even a near-optimal, solution also grows non-polynomially. This suggests that required are ways to decompose this model onto smaller sub-models which, in turn, leads to a wider research question of decomposing (or *modularizing*) probabilistic KBs. Our results guarantee that even decomposing a KB onto a small number of relatively large modules, i.e., around 1000 constraints each, will lead to a dramatic increase of scalability.

Chapter 7

Application Performance Evaluation

This chapter describes the performance evaluation of P-$\mathcal{SROIQ}$ reasoning algorithms that has been carried out using real application data rather than artificially generated probabilistic knowledge bases. Section 7.1 finishes the performance and scalability experiments with the PSAT algorithm by evaluating it on probabilistic ontology mapping data provided by the Ontology Alignment Evaluation Initiative. Section 7.2 is dedicated to evaluation of the Diagnosis and TLexEnt algorithms on a probabilistic formalization of the CADIAG-2 knowledge base.

7.1 Ontology Alignments Validation

This section presents evaluation of the PSAT algorithm on knowledge bases which resulted from probabilistic formalization of uncertain ontology alignments. The formalization itself is presented in Section 3.3. The purpose of these experiments is twofold:

- First, probabilistic validation of ontology alignments is a realistic use case for probabilistic DLs [29]. As such, our experiments aim to demonstrate that our system can be useful for that task since it can handle reasonably large and naturally occurring collections of probabilistic mappings over real ontologies.

- Second, probabilistic KBs which capture uncertain ontology alignments have certain features that could potentially make them harder for reasoning algorithms than random KBs (in particular, large probabilistic signatures, see below). Therefore, it is important to evaluate how well the algorithm handles such extra complexities.

The second point deserves some additional explanation. Recall from Section 2.3.1 that probabilistic signature is the set of all concepts that appear in probabilistic statements in a PTBox. The size of the signature may have a strong impact on hardness of reasoning, in particular, PSAT, because it determines the upper bound on the number of possible worlds (such number is exponential in the signature size). Therefore large signatures may lead to even larger than normal spaces of columns for the PSAT linear program which, in turn, may have a negative impact on convergence of the column generation process.

Normally, the size of probabilistic signature is expected to be smaller than the number of probabilistic statements. This is the case for the BCRA ontology, the CADIAG-2 KB, as well as in most of the experiments conducted by us and prior to us (see [98, 78, 81]). This is so because normally knowledge engineers make more than one statement about each particular concept. For example, one symptom is typically related to several diseases, each disease is associated with multiple risk factors and so on. This is, however, not the case with probabilistic ontology mappings. Most of the mapping tools output one-to-one correspondences between concept names from both ontologies. Since most of the time the correspondences mean equivalence, it is very rare, if ever, that a tool would map one concept to multiple concepts from another ontology. As a result, each probabilistic statement presents a fresh pair of concepts, which makes the signature as large as the total number of statements.

Finally, we use the Jaccard-based probabilistic formalization of mappings (see Section 3.3) so each conditional constraint involves complex concept expressions. This is also likely to be unusual for hand crafted P-$\mathcal{SROIQ}$ KBs and makes these experiments a distinctive source of evaluation of our PSAT algorithm.

7.1.1 Experimental Setup

One particularly established source of uncertain alignments produced by various tools over real and synthetic ontologies is the Ontology Alignment Evaluation Initiative (OAEI).[1] There are multiple tracks used to evaluate different aspects of ontology matching technologies. For our scalability evaluation experiments we use the Anatomy track dataset from OAEI-2009.[2] The contest consisted of finding alignments between the Adult Mouse Anatomy ontology and a part of the NCI Thesaurus ontology which describes the human anatomy. We pick that dataset because, first, it involves real ontologies, second, it is large so it can push the PSAT algorithm to its limits, and third, due to its size a complete reference alignment is unavailable (as of 2009), so the need of automated validation is especially acute.

[1] `http://oaei.ontologymatching.org`

[2] `http://oaei.ontologymatching.org/2009/anatomy/`

The track consists of four tasks but only the first is mandatory for all participating tools. In that task each tool is applied with standard settings and attempts to produce the best alignment without any extraneous help, such as a partial alignment. A number of tools have participated but not all of them produced uncertain mappings. We picked the alignments for those tools which generate confidence values: AgrMaker, Aroma, ASMOV, DSSim, kosimap, and Lily (references can be found in the final report [53]).

The alignments are specified in the Alignment format[3] expressed in RDF/XML, which can be straightforwardly translated into P-$\mathcal{SROIQ}$ syntax. As mentioned earlier we use the Jaccard translation function (see Section 3.3) so each uncertain mapping is translated into a single conditional constraint. For example, the fragment shown below specifies that the Mouse Anatomy entity `MA_0002401` is equivalent to the NCI entity `NCI_C52561` with confidence of 0.854.

```
<map>
  <Cell>
    <entity1 rdf:resource=``http://mouse.owl#MA_0002401''/>
    <entity2 rdf:resource=``http://human.owl#NCI_C52561''/>
    <measure rdf:datatype=``xsd:float''>0.854</measure>
    <relation>=</relation>
  </Cell>
</map>
```

The corresponding conditional constraint is
`(MA_0002401⊓NCI_C52561|MA_0002401⊔NCI_C52561)[0.854,0.854]`.

The translated alignments appeared too large for the current algorithm, for instance, AgrMaker generates over 1400 uncertain mappings for Task 1. Therefore, we use random samples of the full alignments. The samples are of variable size and, as in other experiments, we run the PSAT algorithm on 10 different samples of each size.

7.1.2 Results and Discussion

The results are presented in Table 7.1. All samples turned out to be probabilistically coherent except of those based on Aroma's alignment, in which the conflicts are rather trivial and caused by the following meaningless mappings:
`(SynonymType|SynonymType)[0.95,0.95]`,
`(ObsoleteClass|ObsoleteClass)[0.95,0.95]`,
`(Subset|Subset)[0.63,0.63]`.

There are few important things to notice in the results. First, the algorithm exhibited a substantial performance variability across different tools but not across different samples of the same tool. In particular, samples from the alignment generated by

[3] `http://oaei.ontologymatching.org/2009/align.html`

Table 7.1: PSAT performance when validating probabilistic ontology alignments

Matching tool	PTBox size	Total time (s)	CG Total (s)	# columns
AgrMaker	250	258	201.2	117
	500	612	588.3	319
	750	1095	911.6	566
Aroma	250	127	101.1	201
	500	585	501.2	650
	750	1294	118.9	1036
ASMOV	250	54	45.7	51
	500	407	388.0	422
	750	1123	1044.2	536
DSSim	250	239	209.4	212
	500	699	613.5	488
	750	1304	1199.2	752
kosimap	250	45	39.2	45
	500	164	141.3	71
	750	361	332.6	122
Lily	250	112	88.5	67
	500	399	366.2	112
	750	674	603.7	285

kosimap and Lily appear to be considerably easier than others. This phenomenon deserves a careful future investigation because given that the classical part of the KBs was kept fixed, it may provide insight into why some collections of conditional constraints are harder than others. At this point it is unclear if the variability is due to the nature of our particular algorithm or there are more fundamental reasons.

Second, the results show that signature can indeed be problematic in two respects. Convergence rate is slower on average than in synthetic PSAT experiments, especially those reported in Section 6.5. In addition, large signatures also lead to a high number of variables in the column generation model (the MILP program (5.4)) which makes it harder for MILP reasoners as can be seen in the CG Total column. Note that both MA and NCI ontologies are fairly axiomatically weak, so their probabilistic samples have low subsumption density. A higher density is likely to improve convergence but can further complicate the CG model and increase the number of iterations of Algorithm 4.

Finally and most importantly, the results demonstrate that Pronto can scale to validate probabilistic alignments of realistic size. Although the full alignments are currently beyond its capabilities, their large portions (over 50%) could be analyzed in reasonable time on a modest hardware. The current implementation of the system could be used for finding incoherences in uncertain alignments even though it does not

implement any specific optimizations for handling large signatures.

7.2 Analysis of CADIAG-2

The probabilistic formalization of CADIAG-2, as described in Section 3.2, is interesting for performance evaluation for several reasons. First (and foremost), it is large and naturally occurring. While propositional and relatively simple, its size makes its analysis a significant challenge. Second, it does contain unsatisfiable fragments which makes it suitable not only for PSAT but also for diagnosis evaluation. Finally, it is good starting point for evaluating feasibility of the TLexEnt algorithm because it also contains contextual conflicts which can be resolved during non-monotonic reasoning.

7.2.1 Performance Metrics

The performance measures collected during PSAT experiments, namely wall time, number of generated columns and the average column generation time, are not sufficient for diagnosis and TLexEnt evaluation because they are not informative enough. For example, it is unclear whether a high number of columns is due to some especially hard PSAT instance solved inside the diagnosis algorithm or due to a high number of conflicts and, consequently, high number of PSAT instances to repair them. Similarly, the measures do not allow us to see a high level picture of what is happening during the TLexEnt algorithm which goes through two main phases: computing lex-minimal subsets of conditional constraints (Phase I) and solving a TLogEnt instance for each of them (Phase II). Specifically, it is important to understand whether a particularly difficult TLexEnt instance is hard because of non-monotonicity, because of difficult linear optimization, or both.

Therefore we collect the following additional performance measures for this section's experiments:

Number of conflicts, N_{IIS} This metric represents the number of conflicts, or, equivalently, the number of irreducibly infeasible systems (IIS, see Section 5.2.1), computed by the Diagnosis algorithm or by the TLexEnt algorithm during Phase I.

Number of repairs, N_R It complements the previous metric by representing the number PSAT tests that need to be made to complete the conflict discovery process, i.e. to prove that no other conflicts exist. Depending on overlap between conflicts the number of repairs can vary widely.

Number of lex-minimal subsets, N_{lex} It represents the total number of lexicographically minimal models.

Time to compute lex-minimal subsets, T_{lex} It measures the total time of Phase I of the TLexEnt algorithm.

TLogEnt time, T_{log} It measures the total time of Phase II of the TLexEnt algorithm.

All time measures are wall time, not CPU time. As in previous experiments all measures are collected by Pronto's internal telemetry mechanism.

7.2.2 Finding Inconsistent Fragments

Finding all minimal inconsistent sets of rules in CADIAG-2 KB was a long standing challenge. In this section we present the values of the metrics gathered in the process of finding conflicts in fragments of CADIAG-2 according to the methodology presented in Section 3.2. As explained in that section we use a slightly relaxed version of the KB which was decomposed onto independent fragments. In total we obtained 1007 fragments with the mean size of 21 probabilistic statements and the maximum size of 134 statements.[4] Each size includes the full classical part of CADIAG-2 KB (i.e., the full hierarchies of symptoms and diseases).

The full diagnosis of all fragments of CADIAG-2 KB was performed in *less than 3.5 hours* and, as mentioned in Section 3.2, resulted in 695 unique minimal unsatisfiable sets of constraints. Figures 7.1, 7.2, and 7.3 show the average time of the diagnosis algorithm plotted against the fragment size (measured in the number of constraints), the total number of conflicts found in the fragment, and the number of repairs in the fragment respectively. The figures expectedly demonstrate that the average diagnosis time grows as a function of all three metrics albeit in different ways.

Figure 7.1 shows a steep growth at about 100 conditional constraints with many outliers. This is due to the fact that larger fragments are more likely to contain unsatisfiable fragments. Those fragments whose conflict density, which can be defined as the number of conflicts per size, is either higher or lower than the average represent the outliers.

Figure 7.2 demonstrates that the diagnosis time indeed grows as a function of the number of conflicts, but the interesting fact is that it does *not* seem to be monotonic, at least for CADIAG-2. After a certain number of conflicts, which in this case is around 7, the average time starts to decrease. A likely explanation is that the total number of PSAT tests needed to find all conflicts, i.e. the number of repairs, does not *necessarily* increase with the number of conflicts. Newly found conflicts can overlap

[4]It might appear that the fragments are too small to be interesting. Indeed, they are trivial *for PSAT* but the Diagnosis problem is substantially more difficult. Figure 7.2 shows that hundreds of PSAT tests are required for some fragments.

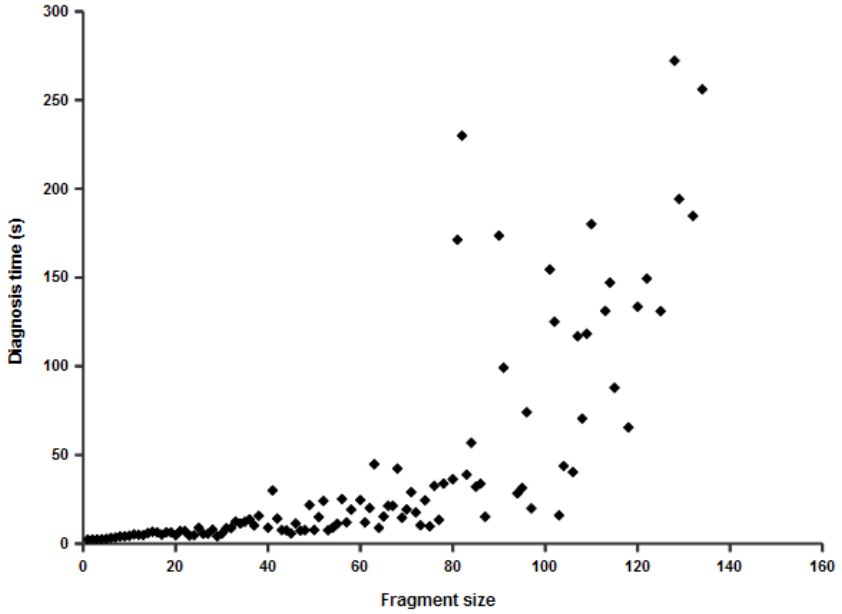

Figure 7.1: The average running time of the diagnosis algorithm against the size of CADIAG-2 fragments.

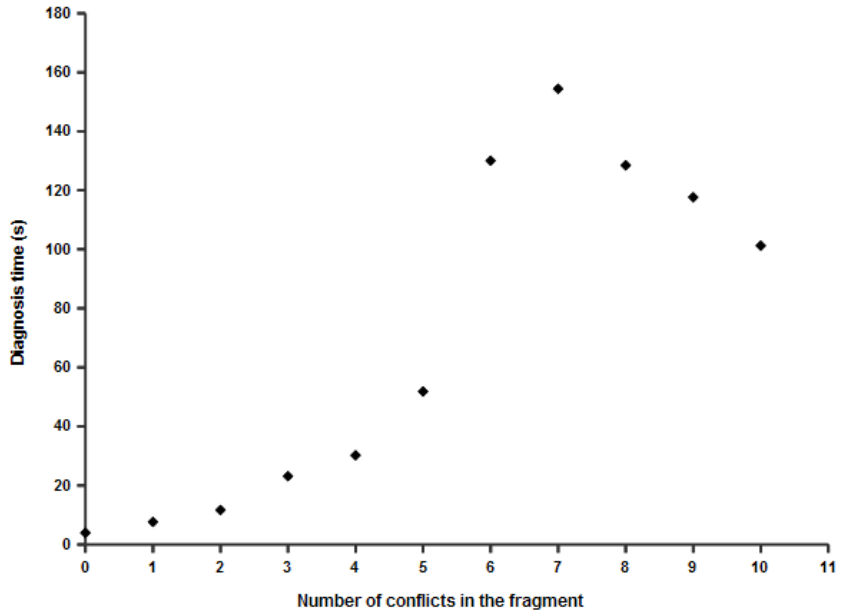

Figure 7.2: The average running time of the diagnosis algorithm against the number of incoherent subsets in CADIAG-2 fragments.

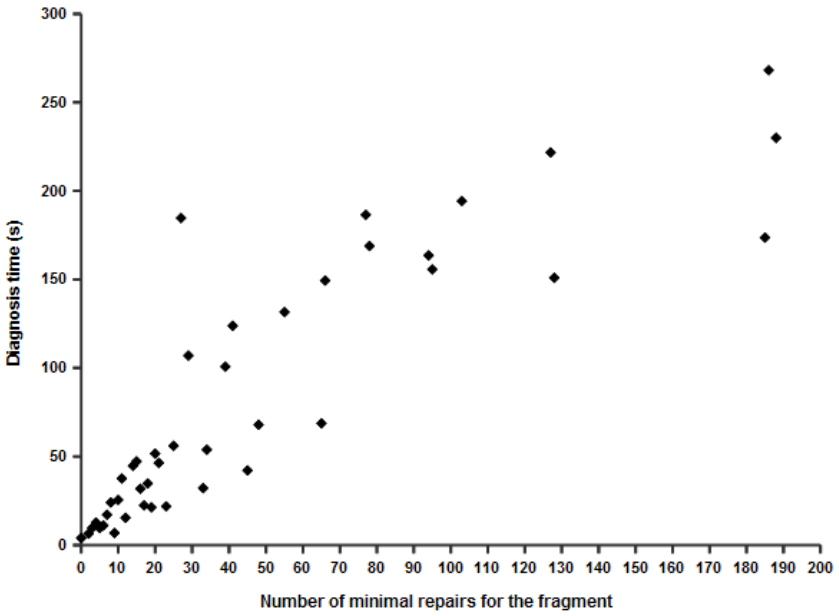

Figure 7.3: The average running time of the diagnosis algorithm against the number of repairs in CADIAG-2 fragments.

with the previous ones such that the number of repairs can decrease which improves the diagnosis performance.

Finally, Figure 7.2 shows that the time monotonically increases with the number of different ways to repair an incoherent fragment. The reason is clear: every repair requires a PSAT test. The outliers are caused by variability of PSAT performance on different subsets of conditional constraints (i.e. some PSAT subsets can be substantially harder than others which causes some diagnoses to take longer even though there are fewer repairs involved).

These experiments show a good robustness of the diagnosis algorithm which can handle quite highly unsatisfiable KBs with up to 10 minimal unsatisfiable subsets. However, more unsatisfiable (or incoherent) KBs could be required to draw more certain charts and fit predictive curves.

7.2.3 Probabilistic Consistency Evaluation

Performance evaluation of the PTCon algorithms described in Section 5.3 was also done on fragments of CADIAG-2 but with three important differences from the previous subsection. First, small fragments have been clumped together so that all of CADIAG-2 was decomposed onto 113 probabilistic KBs, each of which contained between 150 and

250 conditional constraints.[5] Second, the fragments did not contain the unconditional constraints needed to test probabilistic coherence. Third, the fragments were *repaired* in order to produce probabilistically consistent P-$\mathcal{SROIQ}$ KBs.

Due to the numerous probabilistically incoherent fragments of CADIAG-2 (see Section 3.2.3) almost every fragment turns out to be probabilistically inconsistent. Recall from definitions 2.15 and 2.16 that a satisfiable KB is probabilistically inconsistent when the z-partition cannot be constructed, for example, if there are two conditional constraints which do not tolerate each other. The following theorem explains why this is often the case with CADIAG-2:

Theorem 7.1. *Any probabilistically incoherent PTBox $(\mathcal{T}, \mathcal{P})$ such that all constraints have the same evidence concept C is probabilistically inconsistent.*

Proof. $(\mathcal{T}, \mathcal{P} \cup \{(C|\top)[1,1]\})$ is unsatisfiable (by the definition of incoherence), so no constraint in $\mathcal{P}$ is tolerated by $\mathcal{P}$ under $\mathcal{T}$. In other words, there are no "most generic constraints" in $\mathcal{P}$ so the PTBox is inconsistent. □

Most of conflicting subsets reported in Section 3.2.3 involve only a single evidence concept, so it can be concluded that their corresponding fragments are inconsistent.

It is substantially more interesting to evaluate the PTCon algorithms on consistent PTBoxes because then it will have to fully construct the z-partition or the conflict graph. Therefore we repaired all inconsistent fragments in a *random* way. Specifically, a random minimal repair (i.e. a set of constraints which intersects with all conflicts) was removed from each of the fragments. Obviously this may not be the best way of repairing from the domain perspective but it is suitable for the evaluation purposes.

Finally, we have evaluated two PTCon algorithms on all 113 repaired fragments of 150–250 constraints under the time limit of 30 minutes for each fragment. The results are presented in Table 7.2. As you can see the optimized version of the original algorithm was able to solve PTCon for far more fragments and has a better average running time. Unsurprisingly the second algorithm has not been able to compute the conflict graph for any fragment on which the first algorithm ran out of time.

However, the direct comparison was *not* the purpose of this evaluation since the conflict graph based algorithm performs substantially more work (as discussed in Section 5.3). The conflict graph structures that it computes can be further used for TLexEnt, incremental updates to the z-partition, and more. In particular, given a computed conflict graph TLexEnt entailments for PTBox queries (see the next section) reduce to

[5] It would have been ideal to have all fragments of equal size but this is very hard, if not impossible, to guarantee given that some symptoms are associated with many diseases while others to only one disease. Another option, which we considered less interested, was to give up on logical properties of fragments (see Section 3.2.3) and split CADIAG-2 onto random subsets of equal size.

Table 7.2: Performance of PTBox consistency algorithms on fragments of CADIAG-2

	Algorithm 1	Algorithm 2
Total running time (hr)	23.24	40.01
Average time per fragment (s)	845	952
Timeout rate	12.4% (14 out of 113)	38% (43 out of 113)

a series of TLogEnt entailments without the need to compute lexicographically minimal subsets. Furthermore, it does not stop at a point where a partial conflict graph would be sufficient to prove generic inconsistency (i.e., inexistence of the z-partition) but proceeds until conflict sets for all evidence concepts have been computed since they could be used, for example, for repairing the inconsistency. Therefore, even though the second algorithm solves the problem which is harder than necessary for consistency its use may well be beneficial.

7.2.4 Lexicographic Entailment Evaluation

This section presents the final set of experiments aimed to evaluate performance of the TLexEnt algorithm on fragments of CADIAG-2 KB. We use the same repaired fragments as for the PTCon evaluation.

The evaluation of entailment is different from all above presented experiments since it requires generation of queries in addition to knowledge bases. CADIAG-2, as a medical system, enables us to focus on entailments that connect symptoms (or sets of symptoms) to diseases and have a clear medical semantics: "*compute the likelihood of a certain disease given a set of symptoms.*" Therefore query generation amounts to picking symptoms and a disease from the signature of a fragment of CADIAG-2 KB.

We distinguish between PTBox and PABox entailments in this evaluation. For a PTBox $PT = (\mathcal{T}, \mathcal{P})$) a PTBox query consists of a pair of $\mathcal{SROIQ}$ concepts C, D. According to Definition 2.18 its answering requires to compute the minimum and the maximum of $Pr(D)$ subject to all lex-minimal models of PT that satisfy $\{(C|\top)[1,1]\}$. On the other hand, in case of a PABox query for the same PTBox there is a *set* of unconditional concepts $\mathcal{F} = \{(C_i|\top)[l_i, u_i]\}$, a concept D, and all lex-minimal models of PT must satisfy $\mathcal{F}$. In other words, a PTBox TLexEnt query are a special case of a PABox query where $\mathcal{F} = \{(C|\top)[1,1]\}$. It is anticipated that the size of $\mathcal{F}$ will have a negative impact on performance because it is harder to compute lex-minimal subsets of PT (more subsets of $\mathcal{P}$ will be in conflict with $\mathcal{F}$), so PABox queries should be computationally harder.

The reason we still decided to evaluate PTBox entailments separately was to experiment with their *hardest* representatives. More specifically, evidence concepts for

PTBox queries are selected from those symptoms which occur in the top subset of the z-partition, i.e., occur in the *most specific* conditional constraints. This makes it highly likely that the set of lex-minimal subsets will be non-trivial, in other words, the lexicographic reasoning will go through the conflict resolution process, which is something that may not happen if the evidence concept is picked at random.[6] PTBox entailments are only evaluated on fragments of CADIAG-2 which have more than one subset in the z-partition (i.e., there is some overriding of less specific constraints by more specific ones). On the other hand, PABox queries are generated by picking subsets of symptoms randomly and are evaluated on every fragment. Each PABox query models a situation when a patient comes with a set of symptoms and his or her chances of having a particular disease need to be computed. Conclusion concepts (diseases) for both types of queries are picked randomly as they have no impact on computation of lex-minimal subsets.

We have reused z-partitions generated during the consistency evaluation for entailment experiments. Pronto provides a functionality for storing and loading z-partitions and conflict graphs which is useful when multiple queries are executed against a constant KB. This means that the timing results below do not account for time spent on generic consistency checks.

We first carried out the PTBox entailment evaluation on 72 repaired fragments of CADIAG-2 with a non-trivial z-partition. Under the time limit of 30 minutes Pronto was able to answer all 130 queries. The aggregated performance measures are presented below in the left column of Table 7.3.

Next we conducted the PABox entailment evaluation on all 113 repaired fragments of CADIAG-2. We varied the size of $\mathcal{F}$ between 1 and 5. All probability intervals in $\mathcal{F}$ were set to $[1, 1]$ since, first, this is likely to be a more common use case and, second, it is computationally harder because more conflicts will arise during the lexicographic reasoning.[7] The results are presented in the right column of Table 7.3.

The results show that such measures as the number of conflicts and repairs vary far more widely for PABox entailments which makes reasoning performance much less predictable. Also note that even for relatively small fragments of CADIAG-2 the number of lex-minimal subsets can go above hundred. Therefore the Phase II of the TLexEnt algorithm becomes a significant challenge since in general each subset requires solving a TLogEnt instance.

Finally, Table 7.4 shows detailed information for TLexEnt queries where measures are grouped by PABox size. It could be seen that the number of conflicts, the number

[6] It is not guaranteed though. It may be the case that all constraints with C as the evidence concept are tolerated by all subsets in the z-partition, but this seems to be rare for CADIAG-2.

[7] For any set of conditional constraints $\mathcal{G}$ and any concept C, unsatisfiability of $\mathcal{G} \cup \{C|\top)[l, 1]\}$ for $l < 1$ implies unsatisfiability of $\mathcal{G} \cup \{C|\top)[1, 1]\}$ but not vice versa.

Table 7.3: Performance of the TLexEnt algorithm on PTBox and PABox queries against fragments of CADIAG-2

	PTBox queries	**PABox queries**
Total number of queries	130	360
Total running time (hr)	10.8	73.1
Average time per entailment (s)	301	454
Time-out rate	0%	20.2% (73/3600)
Average number of conflicts	4.4	5.4
Max number of conflicts	22	105
Average number of repairs	4.8	8.7
Max number of repairs	22	233
Average number of lex-minimal subsets	1	2.9
Max number of lex-minimal subsets	2	192

of repairs, the average time and the time-out rate all rise with the size of PABox. This data supports our initial hypothesis that PABox entailments tend to get harder as the number of probabilistic assertions grows and that trend cannot be attributed solely to the growing size of the KB. The real reason is the growing complexity of non-monotonic lexicographic reasoning which goes through a more complicated conflict resolution process (Phase I of TLexEnt). In other words, more contextual conflicts between background, or PTBox, probabilistic knowledge and individual, or PABox, probabilistic knowledge have to be resolved on average as the amount of the latter increases. The last row, which shows somewhat fewer conflicts and repairs than the previous, may appear surprising but it is likely to be due to two reasons: first, the numbers are calculated *only* for queries on which the algorithm was able to terminate[8] and second, the number of TLexEnt instances with 5 PABox constraints is relatively small (because not all fragments of CADIAG-2 contain 5 or more distinct symptoms). Our hypothesis, which is backed by the monotonically increasing time-out rate, is that the results have been slightly skewed by some fragments of CADIAG-2 with unrelated symptoms so that the average number of conflicts appeared to be fewer than expected. At the same time most of the "typical" fragments with PABoxes of 5 constraints were simply to hard for the TLexEnt algorithm to terminate.

Few general conclusions can be drawn from the experimental results described in this Section. First, TLexEnt as a reasoning procedure is indeed much computationally harder than PSAT (at least for the current implementations). This was expectable, of course, but the experiments showed *how big* is the gap in scalability between PSAT

[8]Unfortunately, the current evaluation framework does not allow us to obtain any performance measures when the reasoning algorithm is *forced* to terminate. We plan to address this issue in future versions of Pronto and PREVAL-DL.

Table 7.4: TLexEnt performance measures against the size of PABox

PABox size	Average number of conflicts	Average number of repairs	Average time (s)	Time-out rate
1	1.5	1.6	293	0% (0/72)
2	2.4	3.5	433	11% (7/65)
3	5.9	11.5	500	26% (15/57)
4	11.1	17.9	577	47% (23/49)
5	9.3	13.9	547	64% (28/44)

and TLexEnt. Currently it appears to be about an order of magnitude for PTBox queries and PABox queries for small PABoxes. Second, the results clearly indicate that having scalable PSAT and TLogEnt implementations is not sufficient for scalable lexicographic reasoning because the *number* of PSAT and TLogEnt, which depends on the number of contextual conflicts and their overlap (conflict landscape), grows super-polynomially. Finally, CADIAG-2 fragments exhibited substantial variability in the number of conflicts which means that more evaluation data is required. The conflict landscape is the key parameter for assessing practicality of the current (and likely future) implementations so it is very important to determine it for realistic knowledge bases. Fortunately, the current scalability limits, i.e. 150–250 probabilistic statements, make P-$\mathcal{SROIQ}$ a feasible formalism for a range of applications (at least from the computational point of view). Therefore more test cases can be produced which, in turn, will provide a better picture of the conflict landscape and will be a source for further optimizations.

Chapter 8

Pronto: A Practical P-$\mathcal{SROIQ}$ Reasoner

This chapter describes the design and architecture of Pronto, a probabilistic reasoner for P-$\mathcal{SROIQ}$ which implements all optimized reasoning algorithms presented in Chapter 5. It is divided into three parts. First, Section 8.1 presents the system overview, the current syntactic representation of probabilistic KBs and provides information about using Pronto, either from the command line or via its application programming interface (API). Section 8.2 describes the major architectural components of the reasoner. Finally, Section 8.3 explains the main configuration options.

8.1 Pronto Overview

Pronto is an open source tool implemented in Java. For the sake of simplicity it does not currently provide any graphical user interface. The only functionality that it provides in addition to the P-$\mathcal{SROIQ}$ reasoning algorithms is loading and serialization of probabilistic KBs.

8.1.1 Knowledge Base Syntax

The syntax of P-$\mathcal{SROIQ}$ is quite simple as it consists only of conditional constraints (PTBox and PABox, see Section 2.3.1). Since an arbitrary OWL 2 ontology can act as a classical part of a probabilistic KB as well as arbitrary concept expressions can appear in conditional constraints, it is natural to use standard OWL syntaxes as a basis for encoding probabilistic KBs. The current implementation of Pronto uses *axiom annotations* to represent probability intervals for both kinds of constraints.

In OWL 2 annotations are the standard mechanism to associate additional information with ontologies, axioms, and entities. Each annotation has an annotation *property*

and an annotation *value*, where the former is an IRI (internationalized resource identifier) specifying the type of the annotation and the latter can be an IRI, an anonymous individual, or a literal.[1] Examples of commonly used annotations are `rdfs:label` and `rdfs:comment` which represent labels and comments attached to the ontology itself or its entities and axioms.

We use the fixed annotation property `pronto#certainty`[2] for probabilistic annotations. The value is always a string literal which represents an interval in the format ``lower bound;upper bound'', where the bounds are string representations of floating point numbers according to the standard lexical rules.[3] PTBox constraints of the form $(D|C)[l,u]$ are encoded as probabilistically annotated TBox axioms $C \sqsubseteq D$ while PABox axioms of the form $(C|\top)[l,u]$ for a as ABox axioms $C(a)$ with the same kind of annotation. Here are the examples of PTBox and PABox constraints respectively in the OWL/XML syntax (observe that the first example contains a complex concept as the conclusion in the conditional constraint):

```
<SubClassOf>
  <Annotation>
    <AnnotationProperty IRI=''pronto#certainty''/>
    <Literal>0.7;0.95</Literal>
  </Annotation>
  <Class IRI=''Bird''/>
  <ObjectIntersectionOf>
    <Class IRI=''FlyingObject''/>
    <Class IRI=''WingedObject''/>
  </ObjectIntersectionOf>
</SubClassOf>


<ClassAssertion>
  <Annotation>
    <AnnotationProperty IRI=''pronto#certainty''/>
    <Literal>0.7;0.95</Literal>
  </Annotation>
  <Class IRI=''Bird''/>
  <NamedIndividual IRI=''Tweety''/>
</ClassAssertion>
```

A probabilistic KB can be represented as either a single OWL document in which classical and probabilistic statements are mixed together or as an OWL document with only probabilistic statements which imports one or more classical OWL ontologies. In

[1] The specification is available at http://www.w3.org/TR/owl2-syntax/#Annotations.
[2] The full IRI is `http://clarkparsia.com/pronto#certainty`.
[3] See, e.g., 3.2.4.1 in `http://www.w3.org/TR/xmlschema-2#float`.

the first case, once the document is parsed by an OWL parser, Pronto must separate the classical and the probabilistic parts based on annotations. In the second case, the classical part is simply the import closure of the main (i.e., probabilistic) ontology. The second option is preferred because it highlights the separation between classical and probabilistic knowledge and also enables the classical part to evolve as an independent ontology.

Annotation-based encoding has several advantages, for example, there is no need to extend low-level parsers because probabilistic KBs are represented as syntactically well-formed OWL ontologies. A P-$\mathcal{SROIQ}$ parser can therefore work on a higher level by analyzing annotations of already parsed axioms. Another advantage is that all standard OWL syntaxes become available "for free" by virtue of already existing parsers. Furthermore, Pronto does not even need to analyze which of them is being used for a particular KB (this is done by one of available APIs, e.g., the OWL API[4]).

The disadvantages are also important. First, annotations are quite limited which can create problems if P-$\mathcal{SROIQ}$'s syntax is to be extended, for example, to support probabilistic independence or qualitative constraints. Second, annotations have been designed for adding information that has no formal semantics. Therefore, any application that does not understand a particular annotation is free to drop it and interpret the annotated axiom as normal. There is currently no way to indicate that a tool should ignore axioms with annotations that it does not understand. As a consequence, Pronto must remove probabilistically annotated axioms when assembling the classical part since otherwise they will be treated as normal DL axioms. Such issues suggest that in the future it might be beneficial to extend one of OWL syntaxes to accommodate probabilities (that, of course, will require extensions to the parser).

8.1.2 Command Line Usage and API

Pronto provides very simple interfaces for users (via command line) and applications (via an API). The main class for command line usage is `com.clarkparsia.pronto.Pronto` while the main interface in the API is `com.clarkparsia.pronto.ProntoReasoner`. The set of reasoning services accessible via both interfaces is largely the same and the correspondence is presented in Table 8.1.

The API provides access to few other features, in particular, computing TLogEnt and all minimal incoherent fragments.

[4] `http://owlapi.sourceforge.net/`

Table 8.1: The correspondence between Pronto's command line arguments and methods of the API.

Reasoning task	**Command line arguments**	**Method in** `ProntoReasoner`
PSAT	`-psat`	`isSatisfiable(...)`
PTCon	`-consistency`	`isConsistent(...)`
TLexEnt $(D\|C)[?,?]$	`-entail` `<evidence IRI>` `<conclusion IRI>`	`subsumptionEntailment(...)`
TLexEnt $(C\|\top)_o[?,?]$	`-entail` `<individual IRI>` `<conclusion IRI>`	`membershipEntailment(...)`
Diagnosis	`-unsat_subsets`	`computeMinUnsatisfiableSubsets(...)`

8.2 Architecture

Pronto has a layered architecture, presented in Figure 8.1. Each layer has one or more components which invoke other components at the same level or at the next lower level. Lower level components never invoke upper level components but simply pass the requested information upwards.

8.2.1 Linear Program Layer

The main function of the components at the lowermost level is managing linear programs which are optimized in order to solve PSAT and TLogEnt problems. As mentioned earlier, these linear programs usually have exponentially many variables so it is futile to try to represent them explicitly. The linear program manager (LPM) and the column generator (CG) collectively implement the column generation algorithm described in Section 5.1 while other components, namely the various LP/MILP solvers and the DL reasoner provide the necessary optimization and $\mathcal{SROIQ}$ reasoning services.

The LPM is responsible for producing the initial version of the restricted master program (5.5), incorporating each new column into it, and checking the optimality (i.e. stopping) criteria. It interacts with a simplex solver, for example GLPK or CPLEX, which solves the current program (2.3) and returns its primal and dual solutions. The latter is supplied to the CG component in order to guide its search for a new, improving column. The LPM is also responsible for stabilization of the linear program and triggering early unsatisfiability detection checks (see Section 5.1.5).

The CG component implements Algorithm 4. It initializes and maintains the binary linear program (5.4), accepts the dual values u^T from the LPM, and generates new

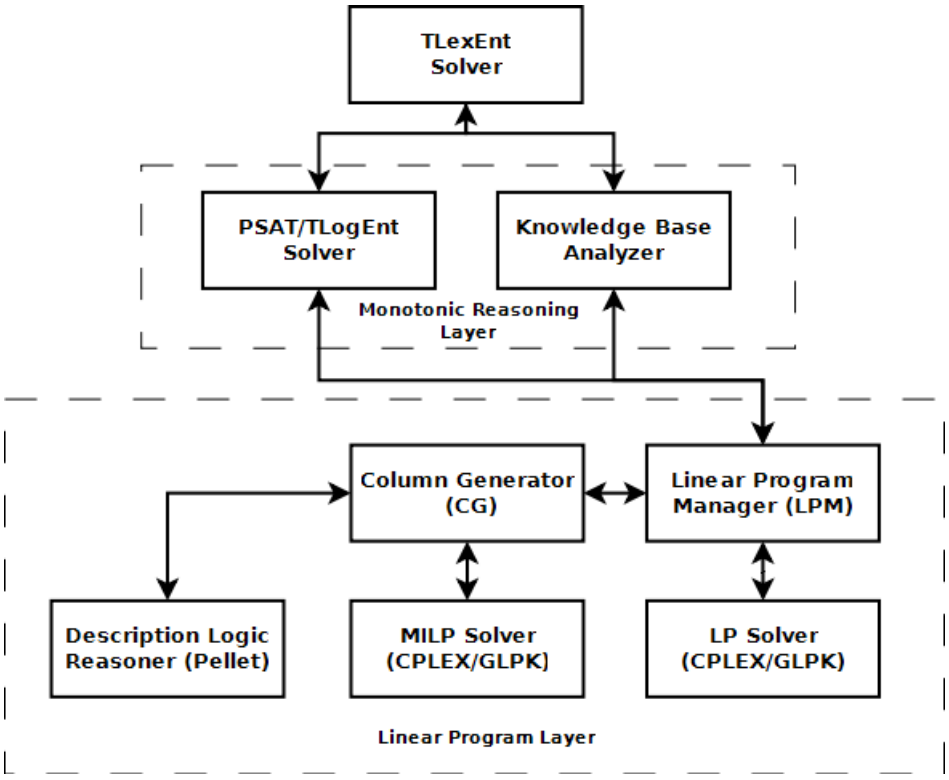

Figure 8.1: The layered architecture of Pronto

column candidates using a MILP solver. It then interacts with Pellet to check validity of each candidate and, if found valid, passes it back to the LPM. This component implements a number of optimizations described in Section 5.1.5 such as exploiting the concept hierarchy, multiple columns generation, optimistic generation and so on.

Currently, Pellet is the only $\mathcal{SROIQ}$ reasoner that Pronto can interact with. However, this is planned to be refactored to introduce an abstract interface, e.g., OWL API, between CG and the reasoner. This would allow us to use other reasoners, such as FaCT++, HermiT, or RACER, as well as specialized reasoners for particular profiles of OWL 2 or other logics.

8.2.2 Monotonic Reasoning Layer

The components on the next layer use the underlying linear programs to solve PSAT and TLogEnt, and also analyze probabilistic KBs. The first two tasks are straightforward. They amount to checking if the generated linear program (5.5) has the optimal objective value less than 1 (PSAT) or optimizing it in both directions (TLogEnt).

The analyzer implements algorithms 5 and 6 for solving the Diagnosis problem (see Definition 5.5). This reasoning service is important by itself as well as for lexicographic entailment. The current implementation closely interacts with the LPM to perform a

trial-and-error relaxation of the linear program (5.5) in order to discover all irreducible subprograms (IIS) which optimal value is less than 1 (they correspond to minimal conflicts). A future implementation may transform the main program (5.5) into another linear program which solutions correspond to the set of IISes of the original program (see [66] for details).

8.2.3 Non-monotonic Reasoning Layer

The uppermost layer consists of a single component: the lexicographic reasoner. It implements the TLexEnt algorithm 9 which relies on the KB analyzer and the TLogEnt solver. TLexEnt is equivalent to solving TLogEnt for all lexicographically minimal subsets of the KB [136]. The latter require the auxiliary data structure called *z-partition* (see Section 2.3.2). The component implements two algorithms to compute z-partition: the optimized Lukasiewicz's algorithm 7 and the Diagnosis-driven algorithm 8. Pronto provides configuration options to choose one of them depending on the input KB.

8.3 Configuring Pronto

A few aspects of Pronto's behavior can be controlled via constant members of the class `com.clarkparsia.pronto.Constants`. Most importantly Pronto can be configured to use different external packages for solving LP and MILP instances (options `Constants.LP_SOLVER` and `Constants.MIP_SOLVER`). Each package should be wrapped into a class implementing the common `com.clarkparsia.pronto.lp.LPSolver` and `com.clarkparsia.pronto.lp.MIPSolver` interfaces. Currently Pronto supports two such packages: the GNU Linear Programming Kit (GLPK) and IBL CPLEX. Other configuration options can be used to set the probability threshold to test coherence (option `Constants.COHERENCE_THRESHOLD`), switch between the two PTCon algorithms (Algorithm 7 and Algorithm 8, option `Constants.USE_CG_ZPARTITIONER`), and switch on/off column generation for PSAT (option `Constants.PSAT_SOLVER_CLASS`).

In the future, we plan to improve on current configuration capabilities in several directions. First, Pronto can be made to work with different $\mathcal{SROIQ}$ reasoners, not just Pellet, analogously to LP/MILP solvers. Second, other aspects, e.g., main optimizations of the PSAT algorithm, should also be accessible via external configuration. Finally, we intend to provide a more user friendly interface for configuration.

Chapter 9

Conclusion

This chapter concludes the thesis with a summary of the work and results presented in the previous chapters. It discusses significance of this thesis' contributions in the area of managing uncertainty in DL ontologies, reviews the open issues, and finally lists a few directions for future research in the area.

9.1 Summary of Contributions

Our main contributions can be classified into three major categories. First, we have developed and evaluated novel P-$\mathcal{SROIQ}$ reasoning algorithms which can be used for probabilistic ontologies of realistic size. The second line of work was dedicated to the analysis of theoretical properties of P-$\mathcal{SROIQ}$. Third, we have investigated applicability of the logic to modeling knowledge in three different areas (and have been able to solve the long standing problem of finding conflicts in large medical system). *Our major conclusion* is that P-$\mathcal{SROIQ}$ is a computationally practical probabilistic language. Furthermore, while we have identified a set of important limitations of P-$\mathcal{SROIQ}$ from a modeling perspective, the logic proved suitable for probabilistic analysis of various knowledge bases with uncertainty.

9.1.1 Practical Reasoning Algorithms and Evaluation

The importance of *practical* reasoning algorithms is hard to overestimate for KR formalisms. Before our work on P-$\mathcal{SROIQ}$ there existed only proof-of-concept implementations which had not been successfully applied to probabilistic KBs with more than 10—20 probabilistic statements. We pushed the scalability limit two orders of magnitude forward.

Column Generation for PSAT/TLogEnt The central part of this thesis, presented in Section 5.1, is the development of a new hybrid iterative procedure for solving probabilistic satisfiability (PSAT) (see algorithm 3) and tight logical entailment (TLogEnt) problems (see algorithm 4). Although column generation has been applied before to propositional PSAT, our work, to the best of our knowledge, is the first to employ it in non-propositional, more precisely DL, settings.

The key idea which enabled column generation for supra-propositional PSAT was to separate $\mathcal{SROIQ}$ reasoning from mathematical optimization. The latter is used to generate an improving column candidate, that is the one which moves the PSAT linear program (5.5) closer to optimality, while the former is necessary for checking its validity with respect to the classical knowledge. That separation, which makes the procedure *hybrid*, is not mandatory for propositional PSAT because propositional reasoning is reducible to MILP (mixed integer linear programming) [84] but is essential for P-$\mathcal{SROIQ}$. Another important difference from propositional PSAT solvers is that our algorithm *iteratively* computes constraints that define the search space for valid columns. Iterativity helps to ignore irrelevant classical formulae since the algorithm may terminate before they first play any role in the column generation process.

We performed an extensive evaluation of our PSAT algorithm on both synthetic and naturally occurring KBs (chapters 6 and 7 respectively). The primary goal of the experiments was to demonstrate that the algorithm scales robustly up to 1000 conditional constraints and beyond. The main synthetic experiments are the following:

- *Propositional PSAT.* The first series of experiments showed that despite its hybridness and generality (with respect to the expressivity of the classical part) the algorithm efficiently deals with propositional KBs. In fact, its performance compares favorably to that of propositional PSAT solvers, especially in relation to the number of generated columns. Therefore, the hybrid approach is a useful technique even in the purely propositional case, especially, if Pellet was substituted by a specialized propositional SAT solver. The results are to some extent due to the effectiveness of the propositional absorption optimization which allows translation of propositional axioms into linear inequalities.

- *Bayesian PSAT.* Next, we evaluated the algorithm on approximate translations of Bayesian networks into P-$\mathcal{SROIQ}$.[1] Such translation can be important for validation of Bayesian networks using terms in context of OWL ontologies. The experiment showed two facts: i) the algorithm can handle networks of approximately 100 nodes and 180—250 links and ii) it is insensitive to tree width of the networks, which is in sharp contrast with Bayesian inference algorithms.

[1] The translation is approximate because it neglects independence constraints.

- *Non-propositional PSAT.* For the final experiment we selected seven real OWL ontologies and augmented them with randomly generated probabilistic statements. The experiment shows that the algorithm scales robustly with respect to such parameters as the number of constraints, size of the signature, and the proportion of unconditional statements. Finally, we demonstrated that it takes substantially less time to prove probabilistic unsatisfiability than satisfiability.

For naturally occurring KBs we used uncertain ontology alignments produced in the Anatomy track of the Ontology Alignment Evaluation Initiative (OAEI) contest in 2009. The alignments were formalized as P-$\mathcal{SROIQ}$ KBs as explained in Section 3.3. The resulting KBs proved to be especially hard due to large probabilistic signatures and the use of complex concepts in conditional constraints. However, Pronto was still able to validate alignments of up to 750 mappings produced by six different tools.

In addition to scalability with respect to the number of probabilistic statements, the PSAT algorithm exhibits relative insensitivity to the amount of the classical knowledge. This is especially visible in experiments with large real ontologies and ontology alignments. Thus, it is fair to say that P-$\mathcal{SROIQ}$ (and Pronto) allow modelers to take *any* OWL DL ontology and enrich it with probabilistic knowledge. The only requirement is that modern DL reasoners, e.g., Pellet, should be capable of performing classical reasoning on the ontology.[2]

Diagnosis, Consistency, and Lexicographic Entailment In Section 5.2 we investigated a new reasoning task for P-$\mathcal{SROIQ}$, namely, the *Diagnosis* problem, to compute all minimal unsatisfiable subsets of probabilistic statements in a unsatisfiable KB. It is useful on its own as well as for analyzing probabilistic incoherence, computing z-partition, and lexicographic minimal sets.

The problem is reducible to a number of PSAT tests, and we developed an efficient algorithm firmly based on certain properties of linear programs (see Theorem 5.2). Our empirical evaluation is based on fragments of the CADIAG-2 KB, many of which are unsatisfiable. We have not done a synthetic evaluation because the number and the overlap between conflicts in realistic KBs—the major impact factors for the performance—remain unclear at this point and so is the methodology for generating test data. Not surprisingly Diagnosis is practically harder than PSAT but our algorithm can still solve it for fragments of CADIAG-2 (including those with numerous conflicts) of approximately 160 statements.

The other developed algorithms are the two z-partition algorithms and the TLexEnt algorithm. The latter has especially strong advantages comparing to the naive

[2] A classical example of an ontology that is (as of 2010) too hard for Pellet is the Family ontology: `www.cs.man.ac.uk/~stevensr/ontology/family.rdf.owl`

Lukasiewicz algorithm (see [136]) because of its direct, Diagnosis-driven search for lexicographically minimal sets. All three algorithms have been evaluated on fragments of CADIAG-2 and demonstrated scalability to over a hundred of probabilistic statements. We must note, however, that the performance is less predictable than that of PSAT because relatively few non-overlapping conflicts can cause them to run an exponential number of PSAT tests. This effect is especially visible in the TLexEnt evaluation on PABoxes of growing size (see Table 7.4).

9.1.2 Analysis of P-$\mathcal{SROIQ}$

While P-$\mathcal{SROIQ}$ offers smooth integration with OWL it has few restrictions which seem strange at first, namely, the separation of classical and probabilistic individuals and the limited support of probabilistic role assertions. No previous work offered a clear explanation of such limitations. Our first contribution to this issue was the translation of P-$\mathcal{SROIQ}$ into a better investigated first-order logic of probability (FOPL) [76, 12]. We showed that the monotonic fragment of P-$\mathcal{SROIQ}$ correspond to a fragment of FOPL with particular, subjective semantics known as FOPL_{II}. Furthermore, the translation allowed us to view P-$\mathcal{SROIQ}$ KBs as standard sets of FOPL formulae rather than a collection of theories. Applying what is known about FOPL_{II} led us to the following conclusions:

- PTBox constraints in P-$\mathcal{SROIQ}$, which may seem as an ideal tool for representing statistical relationships, do not have statistical nature. Instead, they are most reasonably interpreted as beliefs about random individuals.
- P-$\mathcal{SROIQ}$ does not support probabilistic relational structures because its notion of possible world is effectively propositional. This is highlighted by the fact that our translation of an arbitrary P-$\mathcal{SROIQ}$ KB i) uses only ground probabilistic formulas and ii) only a single individual appears in those formulas. In other words, not only P-$\mathcal{SROIQ}$ statements are degrees of belief but also beliefs about a *single*, yet unnamed, individual.
- Relational structures cannot be supported without substantial changes in P-$\mathcal{SROIQ}$'s semantics, direct inference mechanism, and inference methods.
- P-$\mathcal{SROIQ}$ inherits some inferential weakness of FOPL_{II} due to the cautiousness of purely probabilistic reasoning.

This is not meant to claim that P-$\mathcal{SROIQ}$ is useless. It can be seen as a pragmatic approximation of FOPL_{II} which trades certain semantic power for feasibility of reasoning, including non-monotonic reasoning. However, it is important that the limitations are clear and understandable to modelers.

9.1.3 Applicability of P-$\mathcal{SROIQ}$

We studied the prospects of applying P-$\mathcal{SROIQ}$ to various problems to assess its utility given the above limitations. First, in Section 3.1 we looked at the *breast cancer risk assessment* problem because first, there already exist rich and useful OWL ontologies in that domain (e.g., the NCI Thesaurus) and second, the domain involves a lot of uncertain information. We argues that while P-$\mathcal{SROIQ}$ allows for adding uncertain relationships to those ontologies its inferential weakness poses substantial challenges to using it either as a risk prediction model. However, we pointed out that it might be useful for augmenting the general theory of breast cancer, in particular, for reasoning about research findings.

Second, we applied P-$\mathcal{SROIQ}$ to the problem of finding inconsistencies in CADIAG-2—a large rule-based medical expert system (see Section 3.2). Given the number of rules and the presence of confidence values the full analysis of its consistency has been a long standing unsolved problem. We translated CADIAG-2 KB into P-$\mathcal{SROIQ}$, proved the faithfulness of the translation, and formulated the task as an instance of the Diagnosis problem. Finally, after splitting the KB onto smaller fragments we have been able to extract and classify *all* minimal inconsistent sets of rules.

Finally, we applied Pronto to the problem of validating uncertain ontology alignments. P-$\mathcal{SROIQ}$ is ideally suited for this task because i) schema (i.e., TBox) alignments typically do not involve mapping individuals, ii) uncertain relations (e.g., equivalence) between concepts are easily captured in conditional constraints, and iii) it is vital that the formalism allows for working with large and complex OWL ontologies. Our aim was mostly to show that our algorithms can scale to large mappings. We did so in Section 7.1 by using the data from a recent ontology alignment evaluation contest.

Each of the presented studies required substantial domain-specific investigation of modeling with conditional constraints. This includes dealing with confidence regions in BCRA, equisatisfiability and decomposition of CADIAG-2, and comparison of probabilistic validation of mappings to earlier classical approaches in case of ontology alignments. In these studies P-$\mathcal{SROIQ}$ has demonstrated its usefulness as an underlying formalism but cannot yet be considered a user-friendly tool for end modelers.

9.2 Challenges and Future Directions

The work presented in this thesis can be extended in both theoretical and practical directions. The former involves addressing the current limitations of P-$\mathcal{SROIQ}$ while the latter may focus on further enhancements of scalability and modeling relevant domain problems in P-$\mathcal{SROIQ}$.

Modeling In our view, future work on modeling is absolutely essential for providing input on what extensions are the most desirable and what computational issues are the most prohibitive. We believe that scalability of our algorithms make P-$\mathcal{SROIQ}$ a computationally feasible formalism for modeling experiments. One particular option is to look at some previously formulated *challenge problems* for probabilistic languages and try to formalize them in P-$\mathcal{SROIQ}$.[3] Another possibility is to consider the use cases for uncertainty in Health Care and Life Sciences.[4]

Language Extensions Based on our current modeling experience we believe the following enhancements are desirable if P-$\mathcal{SROIQ}$ is to become a full-fledged probabilistic ontology language:

Relational Structures: Supporting probabilistic role assertions is a must for a probabilistic ontology language but unfortunately it would require deep changes to P-$\mathcal{SROIQ}$'s notion of possible worlds and thus semantics. They should become richer approximations of $\mathcal{SROIQ}$ interpretations rather than simple concept types. Unfortunately that would mean that either we give up on combining different kinds of probabilities and concentrate solely on beliefs (i.e., analogously to [139]) or will require a semantic separation between statistics and beliefs, i.e., a Type III-like model theory. The latter, of course, will require a new direct inference mechanism (see Section 2.2.2). We presented some preliminary work in this direction in [113].

Independence: "A serious weakness of probabilistic logic is that it omits what is probably our richest source of probabilistic knowledge: the independence of events." [5]. Adding independence assertions in P-$\mathcal{SROIQ}$ may have a range of benefits ranging from more accurate modeling and strengthened inference to a more modular knowledge representation [37]. On the other hand, this would lead to non-linear systems of inequalities for PSAT, but there are methods, such as Benders decomposition, to adapt column generation to handle those [5].

Extra Probabilistic Features: Purely probabilistic deduction is often too weak. For example, knowing probabilities of events A and B one cannot uniquely determine the probability of $A \sqcup B$. Extra machinery can be employed to overcome that issue, known as inferential vacuity. One of them is the principle of maximum entropy that plays a key role in Objective Bayesianism [189]. Other features might include support of abductive or inductive probabilistic reasoning.

Further Scalability Improvements Finally, we briefly mention a couple of ways to further improve the current reasoning algorithms:

[3] One such collection of challenge problems, maintained by Manfred Jaeger, can be found at `http://www.cs.aau.dk/~jaeger/plsystems/challenges.html`

[4] `http://www.w3.org/wiki/HCLS/UncertaintyUseCases`

Coherence checking: Pronto requires a better probabilistic coherence testing algorithm which does not depend on the chosen lower probability. We are especially interested in algorithms, which do not require solving sequences of linear programs. One such technique was described in [36] but its practical implementability is yet to be investigated.

Probabilistic Modularity: As pointed out in Section 6.6 a key to more scalable PSAT is a generic procedure for partitioning probabilistic KBs (an *ad hoc* method was very very successful for CADIAG-2). This can be either a high-level procedure for separating conditional constraints or a lower-level algorithm for partitioning linear programs, e.g., Dantzig-Wolfe decomposition [42].

Intelligent Diagnosis: The Diagnosis algorithm can be improved by using methods for discovering all irreducible infeasible subsystems in one go [66, 161]. However, to the best of our knowledge, they have never been made to work with column generation so their practical benefits and potential pitfalls are unknown.

Each of these extensions may turn out to be a significant challenge from both theoretical and practical points of view. However, we believe that a set of scalable core algorithms, described experience with advanced mathematical programming techniques, and a flexible evaluation framework collectively lay out solid foundations for future developments and experimentation.

Appendix A

Proofs of Theorems

Proof of Theorem 3.3 We first prove the following claim: Let $\Phi_{\mathcal{M}}$ be a signature of the mapping and $I_{\mathcal{M}}$ be a set of all *realizable* concept types of T over $\Phi_{\mathcal{M}}$. Then there exists a probabilistic model Pr of PT such that $Pr(I) > 0$ for every $I \in I_{\mathcal{M}}$.

We prove that in a constructive way for $t = t^l_{castano}$ (it can be done analogously for other probabilistic translation functions proposed in Section 3.3.2). The set $I_{\mathcal{M}}$ is finite so let Pr_u be a uniform probability distribution over $I_{\mathcal{M}}$. Let us show that Pr is a model of all conditional constraints of $\{t(m)|m \in \mathcal{M}\}$. Assume it is *not* a model of some constraint $(D|C)[n, 1]$. This means that $Pr_u(D|C) < 1$. Expanding the left handside we get:

$$\sum_{I \models C \sqcap D} Pr_u(I) < \sum_{I \models C} Pr_u(I), \tag{A.1}$$

$$\sum_{I \models C \sqcap D} Pr_u(I) < \sum_{I \models C \sqcap \neg D} Pr_u(I) + \sum_{I \models C \sqcap D} Pr_u(I),$$

$$\sum_{I \models C \sqcap \neg D} Pr_u(I) > 0$$

However, this contradicts the way Pr_u was constructed because T must contain the axiom $C \sqsubseteq D$ (otherwise we would not have the constraint $(D|C)[n, 1]$). Therefore all concept types containing the pair $C, \neg D$ are not realizable. Thus Pr_u is a model of PT and the claim is true.

Now suppose that $PT \models (X|\top)[0, 0]$ for some $X \in \Phi_{\mathcal{M}}$ but $T \not\models X \sqsubseteq \bot$. In that case there have to exist a realizable concept type $I \in I_{\mathcal{M}}$ such that $X \in I$. According to the above claim there have to exist a probabilistic model Pr of PT such that $Pr(I) > 0$. But since $X \in I$ it means that PT cannot entail $(X|\top)[0, 0]$ because Pr_u is a model of PT and not a model of $(X|\top)[0, 0]$. □

Proof of Theorem 5.2 When the column generation process terminates the final RMP has the form :

$$\begin{aligned} \max\ & z = 1x_B \\ \text{s.t. } & Bx_B \leq b \\ & x \geq 0 \end{aligned}$$

and its dual is:

$$\begin{aligned} \min\ & b^T u \\ \text{s.t. } & B^T u \leq 1 \\ & u \geq 0 \end{aligned}$$

The strong duality theorem of Linear Programming [32] states that $1^T x^* = b^T u^* < 0$, where x^* and u^* stand for final optimal solutions of RMP and its dual respectively. Now, consider B_I, i.e., the matrix of the reduced final RMP after dropping those rows from B which correspond to zero components of u^*. The reduced RMP is:

$$\begin{aligned} \max\ & z = 1x_B \\ \text{s.t. } & B_I x_B \leq b_I \\ & x_B \geq 0 \end{aligned}$$

and its dual is:

$$\begin{aligned} \min\ & b_I^T u_I \\ \text{s.t. } & B_I^T u_I \leq 1 \\ & u_I \geq 0 \end{aligned}$$

Now consider u_I^*. It is a solution to the reduced dual of the final RMP and its objective value is $b_I^T\ u_I^*$ is equal to $b^T\ u^*$ so is less than 1. Furthermore, it is the case that $N_I^T\ u_I^* < 1$ (since $N^T\ u^* < 1$) so it is a feasible solution to the dual of the reduced LP with all columns (the full LP). Therefore, from the strong duality theorem it follows that the optimal value of the reduced full LP is strictly less than 1. □

Bibliography

[1] M. Abadi and J. Y. Halpern. Decidability and expressiveness for first-order logics of probability. *Information and Computation*, 112(1):1–36, 1994.

[2] K. Adlassnig. Fuzzy set theory in medical diagnosis. *IEEE Transactions on Systems, Man and Cybernetics*, 16(2):260–265, 1986.

[3] K. Adlassnig, G. Kolarz, W. Effenberger, and H. Grabner. Cadiag: Approaches to computer-assisted medical diagnosis. *Computers in Biology and Medicine*, 15:315–335, 1985.

[4] K. Adlassnig, G. Kolarz, W. Scheithauer, and H. Grabner. Approach to a hospital-based application of a medical expert system. *Informatics for Health and Social Care*, 11(3):205–223, 1986.

[5] K. A. Andersen and J. Hooker. Bayesian logic. *Decision Support Systems*, 11(2):191–210, 1994.

[6] S. Andreassen, M. Woldbye, B. Falck, and S. K. Andersen. MUNIN - a causal probabilistic network for interpretation of electromyographic findings. In *International Joint Conference on Artificial Intelligence*, pages 366–372, 1987.

[7] B. C. Arnold and R. M. Shavelle. Joint confidence sets for the mean and variance of a normal distribution. *The American Statistician*, 52(2):133–140, 1998.

[8] F. Baader, S. Brandt, and C. Lutz. Pushing the EL envelope. In *International Joint Conference on Artificial Intelligence*, pages 364–369, 2005.

[9] F. Baader, D. Calvanese, D. McGuiness, D. Nardi, and P. F. Patel-Schneider. *Description Logic Handbook*. Cambridge University Press, 2003.

[10] F. Baader and B. Hollunder. KRIS: Knowledge representation and inference system. *SIGART Bulletin*, 2(3):8–14, 1991.

[11] F. Bacchus. Lp, a logic for representing and reasoning with statistical knowledge. *Computational Intelligence*, 6:209–231, 1990.

[12] F. Bacchus. *Representing and reasoning with probabilistic knowledge*. MIT Press, 1990.

[13] F. Bacchus. Using first-order probability logic for the construction of Bayesian networks. In *International Conference on Uncertainty in Artificial Intelligence*, pages 219–226, 1993.

[14] F. Bacchus, A. Grove, J. Y. Halpern, and D. Koller. From statistical knowledge bases to degrees of belief. *Artificial Intelligence*, 87(1–2):75–143, 1996.

[15] F. Bacchus, A. J. Grove, D. Koller, and J. Y. Halpern. From statistics to beliefs. In *Advances in Artificial Intelligence Conference*, pages 602–608, 1992.

[16] O. Barnett and J. B. Paris. Maximum entropy inference with quantified knowledge. *Logic Journal of the IGPL*, 16(1):85–98, 2008.

[17] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(5):34–43, 2001.

[18] A. Borgida. On the relationship between description logic and predicate logic. In *International Conference on Information and Knowledge Management*, pages 219–225, 1994.

[19] A. Borgida, R. J. Brachman, D. L. McGuinness, and L. A. Resnick. CLASSIC: A structural data model for objects. In *SIGMOD Conference*, pages 58–67, 1989.

[20] R. J. Brachman, A. Borgida, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Resnick. The CLASSIC knowledge representation system or, KL-ONE: The next generation. In *International Conference on Fifth Generation Computer Systems*, pages 1036–1043, 1992.

[21] R. J. Brachman, V. P. Gilbert, and H. J. Levesque. An essential hybrid reasoning system: Knowledge and symbol level accounts of KRYPTON. In *International Joint Conference on Artificial Intelligence*, pages 532–539, 1985.

[22] R. J. Brachman, H. J. Levesque, and R. Fikes. KRYPTON: Integrating terminology and assertion. In *Advances in Artificial Intelligence Conference*, pages 31–35, 1983.

[23] R. J. Brachman and J. G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, 1985.

[24] J. S. Breese. Construction of belief and decision networks. *Computational Intelligence*, 8:624–647, 1991.

[25] A. Calì, T. Lukasiewicz, L. Predoiu, and H. Stuckenschmidt. Tightly integrated probabilistic description logic programs for representing ontology mappings. In *International Symposium on Foundations of Information and Knowledge Systems*, pages 178–198, 2008.

[26] D. Calvanese, G. D. Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *Journal of Automated Reasoning*, 39(3):385–429, 2007.

[27] D. Calvanese, D. Lembo, M. Lenzerini, and R. Rosati. DL-Lite: Tractable description logics for ontologies. In *Advances in Artificial Intelligence Conference*, pages 602–607, 2005.

[28] R. N. Carvalho, K. B. Laskey, and P. C. Costa. PR-OWL 2.0—bridging the gap to OWL semantics. In *Workshop on Uncertainty Reasoning in the Semantic Web, held at the International Semantic Web Conference*, 2010.

[29] S. Castano, A. Ferrara, D. Lorusso, T. H. Näth, and R. Möller. Mapping validation by probabilistic reasoning. In *European Conference on Semantic Web*, pages 170–184, 2008.

[30] A. Charnes and W. W. Cooper. Programming with linear fractional functionals. *Naval Research Logistics Quarterly*, 9:181–186, 1962.

[31] P. Cheeseman. An inquiry into computer understanding. *Computational Intelligence*, 4:58–66, 1988.

[32] V. Chvátal. *Linear programming*. A Series of Books in the Mathematical Sciences. W.H. Freeman and Company, New York, 1983.

[33] A. Ciabattoni and P. Rusnok. On the classical content of monadic G$^{\sim}$ and its applications to a fuzzy medical expert system. In *International Conference on the Principles of Knowledge Representation and Reasoning*, 2010.

[34] G. Coletti, A. Gilio, and R. Scozzafava. Comparative probability for conditional events: a new look through coherence. *Theory and Decision*, 35:237–258, 1993.

[35] A. M. Collins and M. R. Quillian. Retrieval time from semantic memory. *Journal of Verbal Learning and Verbal Behavior*, 8(2):240–247, 1969.

[36] F. G. Cozman. Algorithms for conditioning on events of zero lower probability. In *International Florida Artificial Intelligence Research Society Conference*, pages 248–252, 2002.

[37] F. G. Cozman, C. P. de Campos, and J. C. F. da Rocha. Probabilistic logic with independence. *International Journal of Approximate Reasoning*, 49(1):3–17, 2008.

[38] F. G. Cozman and R. B. Polastro. Loopy propagation in a probabilistic description logic. In *International Conference on Scalable Uncertainty Management*, pages 120–133, 2008.

[39] P. C. G. da Costa, K. B. Laskey, and K. J. Laskey. PR-OWL: A Bayesian ontology language for the semantic web. In *Workshop on Uncertainty Reasoning in the Semantic Web, held at the 5th International Semantic Web Conference*, pages 23–33, 2005.

[40] C. d'Amato, N. Fanizzi, and T. Lukasiewicz. Tractable reasoning with Bayesian description logics. In *International Conference on Scalable Uncertainty Management*, pages 146–159, 2008.

[41] E. Danna, M. Fenelon, Z. Gu, and R. Wunderling. Generating multiple solutions for mixed integer programming problems. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 280–294, 2007.

[42] G. B. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations Research*, 8:101–111, 1960.

[43] S. de Coronado, M. W. Haber, N. Sioutos, M. S. Tuttle, and L. W. Wright. NCI Thesaurus: using science-based terminology to integrate cancer research results. *Studies in Health Technology and Informatics*, 107(1):33–37, 2004.

[44] R. de Salvo Braz, E. Amir, and D. Roth. A survey of first-order probabilistic models. In *Innovations in Bayesian Networks*, pages 289–317. Springer, 2008.

[45] P. S. de Souza Andrade, J. C. F. da Rocha, D. P. Couto, A. da Costa Teves, and F. G. Cozman. A toolset for propositional probabilistic logic. In *Encontro Nacional de Inteligencia Artificial*, pages 1371–1380, 2007.

[46] C. Del Vescovo, B. Parsia, U. Sattler, and T. Schneider. The modular structure of an ontology: an empirical study. In O. Kutz, J. Hois, J. Bao, and B. Cuenca Grau, editors, *Workshop on Modular Ontologies*, volume 211 of *Frontiers in AI and Applications*, pages 11–24. IOS Press, 2010.

[47] A. Doan, J. Madhavan, R. Dhamankar, P. Domingos, and A. Y. Halevy. Learning to match ontologies on the Semantic Web. *International Journal on Very Large Data Bases*, 12(4):303–319, 2003.

[48] A. Doan, J. Madhavan, P. Domingos, and A. Y. Halevy. Ontology matching: A machine learning approach. In *Handbook on Ontologies*, pages 385–404. Springer, 2004.

[49] P. Domingos, D. Lowd, S. Kok, H. Poon, M. Richardson, and P. Singla. Just add weights: Markov logic for the semantic web. In *International Workshop on Uncertainty Reasoning in the Semantic Web*, pages 1–25, 2008.

[50] P. Domingos and P. Singla. Markov logic in infinite domains. In *Probabilistic, Logical and Relational Learning - A Further Synthesis*, 2007.

[51] O. du Merle, D. Villeneuve, J. Desrosiers, and P. Hansen. Stabilized column generation. *Discrete Mathematics*, 194(1-3):229–237, 1999.

[52] M. Dürig and T. Studer. Probabilistic ABox reasoning: Preliminary results. In *International Workshop on Description Logic*, pages 104–111, 2005.

[53] J. Euzenat, A. Ferrara, L. Hollink, A. Isaac, C. Joslyn, V. Malaisé, C. Meilicke, A. Nikolov, J. Pane, M. Sabou, F. Scharffe, P. Shvaiko, V. Spiliopoulos, H. Stuckenschmidt, O. Sváb-Zamazal, V. Svátek, C. T. dos Santos, G. A. Vouros, and S. Wang. Results of the ontology alignment evaluation initiative 2009. In *Ontology Matching*, 2009.

[54] J. Euzenat and P. Shvaiko. *Ontology Matching*. Springer-Verlag, 2007.

[55] M. Ewertz. Risk of breast cancer in relation to social factors in Denmark. *Acta Oncologica*, 27(6):787–792, 1988.

[56] A. Fijany and F. Vatan. New approaches for efficient solution of hitting set problem. In *Winter International Symposium on Information and Communication Technologies*. Trinity College Dublin, 2004.

[57] N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In *International Joint Conference on Artificial Intelligence*, pages 1300–1309, 1999.

[58] A. M. Frisch and P. Haddawy. Anytime deduction for probabilistic logic. *Artificial Intelligence*, 69:93–122, 1994.

[59] H. Gaifman. Concerning measures in first order calculi. *Israel Journal of Mathematics*, 2:1–18, 1964.

[60] M. H. Gail, L. A. Brinton, D. P. Byar, D. K. Corle, S. B. Green, C. Schairer, and J. J. Mulvihill. Projecting individualized probabilities of developing breast cancer for white females who are being examined annually. *Journal of the National Cancer Institute*, 81(25):1879–1886, 1989.

[61] H. Geffner and J. Pearl. A framework for reasoning with defaults. In H. Kyburg, R. Loui, and G. Carlson, editors, *Knowledge Representation and Defeasible Reasoning*, pages 69–87. Kluwer Academic Publishers, 1990.

[62] G. F. Georgakopoulos, D. J. Kavvadias, and C. H. Papadimitriou. Probabilistic satisfiability. *Journal of Complexity*, 4(1):1–11, 1988.

[63] L. Getoor, N. Friedman, D. Koller, A. Pfeffer, and B. Taskar. Probabilistic relational models. In L. Getoor and B. Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press, 2007.

[64] G. D. Giacomo and M. Lenzerini. TBox and ABox reasoning in expressive description logics. In *International Conference on Principles of Knowledge Representation and Reasoning*, pages 316–327, 1996.

[65] R. Giugno and T. Lukasiewicz. P-$\mathcal{SHOQ}$(D): A probabilistic extension of $\mathcal{SHOQ}$(D) for probabilistic ontologies in the semantic web. In *European Conference on Logics in Artificial Intelligence*, pages 86–97, 2002.

[66] J. Gleeson and J. Ryan. Identifying minimally infeasible subsystems of inequalities. *INFORMS Journal on Computing*, 2(1):61–63, 1990.

[67] J. Golbeck, G. Fragoso, F. W. Hartel, J. A. Hendler, J. Oberthaler, and B. Parsia. The national cancer institute's thésaurus and ontology. *Journal of Web Semantics*, 1(1):75–80, 2003.

[68] M. Goldszmidt and J. Pearl. On the consistency of defeasible databases. *Artificial Intelligence*, 52(2):121–149, 1992.

[69] B. C. Grau, I. Horrocks, Y. Kazakov, and U. Sattler. Modular reuse of ontologies: Theory and practice. *Journal of Artificial Intelligence Research*, 31:273–318, 2008.

[70] O. Gries and R. Möller. Gibbs sampling in probabilistic description logics with deterministic dependencies. In *International Workshop on Uncertainty in Description Logics*, 2010.

[71] B. N. Grosof, I. Horrocks, R. Volz, and S. Decker. Description logic programs: combining logic programs with Description Logic. In *International Conference on World Wide Web*, pages 48–57, 2003.

[72] V. Haarslev and R. Möller. RACER system description. In *International Joint Conference on Automated Reasoning*, pages 701–706, 2001.

[73] V. Haarslev, R. Möller, and A.-Y. Turhan. HAM-ALC. In *International Workshop on Description Logic*, 1998.

[74] P. Haddawy. Generating Bayesian networks from probablity logic knowledge bases. In *Uncertainty in Artificial Intelligence*, pages 262–269, 1994.

[75] T. Hailperin. Probability logic. *Notre Dame Journal of Formal Logic*, 25(3):198–212, 1984.

[76] J. Y. Halpern. An analysis of first-order logics of probability. *Artificial Intelligence*, 46:311–350, 1990.

[77] P. Hansen and B. Jaumard. Probabilistic satisfiability. In *Handbook of Defeasible Reasoning and Uncertainty Management Systems*, volume 5, chapter Algorithms for Uncertainty and Defeasible Reasoning, pages 321–367. Kluwer, 2000.

[78] P. Hansen, B. Jaumard, and M. P. de Aragão. Mixed-integer column generation algorithms and the probabilistic maximum satisfiability problem. *European Journal of Operational Research*, 108(3):671–683, 1998.

[79] P. Hansen, B. Jaumard, M. P. de Aragão, F. Chauny, and S. Perron. Probabilistic satisfiability with imprecise probabilities. *International Journal of Approximate Reasoning*, 24(2–3):171–189, 2000.

[80] P. Hansen, B. Jaumard, G.-B. D. Nguetsé, and M. P. de Aragão. Models and algorithms for probabilistic and Bayesian logic. In *International Joint Conference on Artificial Intelligence*, pages 1862–1868, 1995.

[81] P. Hansen and S. Perron. Merging the local and global approaches to probabilistic satisfiability. *International Journal of Approximate Reasoning*, 47(2):125–140, 2008.

[82] J. Heinsohn. Probabilistic description logics. In *International Conference on Uncertainty in Artificial Intelligence*, pages 311–318, 1994.

[83] J. Hooker. A mathematical programming model for probabilistic logic. Working Paper 05-88-89, Graduate School of Industrial Administration, Carnegie Mellon University, Pittsburgh, PA 15213, 1988.

[84] J. N. Hooker. Quantitative approach to logical reasoning. *Decision Support Systems*, 4:45–69, 1988.

[85] M. Horridge, B. Parsia, and U. Sattler. Laconic and precise justifications in OWL. In *International Semantic Web Conference*, pages 323–338, 2008.

[86] M. Horridge, B. Parsia, and U. Sattler. Explaining inconsistencies in OWL ontologies. In *Scalable Uncertainty Management*, pages 124–137, 2009.

[87] I. Horrocks. Using an expressive description logic: Fact or fiction? In *International Conference on Principles of Knowledge Representation and Reasoning*, pages 636–649, 1998.

[88] I. Horrocks, O. Kutz, and U. Sattler. The even more irresistible $\mathcal{SROIQ}$. In *Knowledge Representation and Reasoning*, pages 57–67, 2006.

[89] I. Horrocks and P. F. Patel-Schneider. Reducing OWL entailment to description logic satisfiability. *Journal of Web Semantics*, 1(4):345–357, 2004.

[90] M. C. Horsch and D. Poole. A dynamic approach to probabilistic inference using Bayesian networks. In *International Conference on Uncertainty in Artificial Intelligence*, pages 155–161, 1990.

[91] J. S. Ide, F. G. Cozman, and F. T. Ramos. Generating random Bayesian networks with constraints on induced width. In *European Conference on Artificial Intelligence*, pages 323–327, 2004.

[92] M. Jaeger. Probabilistic reasoning in terminological logics. In *International Conference on Knowledge Representation and Reasoning*, pages 305–316, 1994.

[93] M. Jaeger. Relational Bayesian networks. In *International Conference on Uncertainty in Artificial Intelligence*, pages 266–273, 1997.

[94] M. Jaeger. Reasoning about infinite random structures with relational Bayesian networks. In *International Conference on Principles of Knowledge Representation and Reasoning*, pages 570–581, 1998.

[95] M. Jaeger. Complex probabilistic modeling with recursive relational Bayesian networks. *Annals of Mathematics and Artificial Intelligence*, 32(1–4):179–220, 2001.

[96] M. Jaeger. Parameter learning for relational Bayesian networks. In *International Conference on Machine Learning*, pages 369–376, 2007.

[97] M. Jaeger. Model-theoretic expressivity analysis. In *Probabilistic Inductive Logic Programming*, pages 325–339, 2008.

[98] B. Jaumard, P. Hansen, and M. P. de Aragão. Column generation methods for probabilistic logic. In *Integer Programming and Combinatorial Optimization Conference*, pages 313–331, 1990.

[99] B. Jaumard, P. Hansen, and M. P. de Aragão. Column generation methods for probabilistic logic. *INFORMS Journal on Computing*, 3(2):135–148, 1991.

[100] B. Jaumard and A. D. Parreira. An anytime deduction algorithm for the probabilistic logic and entailment problems. *International Journal Approximate Reasoning*, 50(1):92–103, 2009.

[101] D. Jovanovic, N. Mladenovic, and Z. Ognjanovic. Variable neighborhood search for the probabilistic satisfiability problem. In *International Conference on Metaheuristics*, pages 557–562, 2005.

[102] C. E. Kahn, L. M. Roberts, K. A. Shaffer, and P. Haddawy. Construction of a Bayesian network for mammographic diagnosis of breast cancer. *Computers in Biology and Medicine*, 27(1):19–29, 1997.

[103] A. Kalyanpur, B. Parsia, M. Horridge, and E. Sirin. Finding all justifications of OWL DL entailments. In *International Semantic Web Conference*, pages 267–280, 2007.

[104] A. Kalyanpur, B. Parsia, E. Sirin, and B. C. Grau. Repairing unsatisfiable concepts in OWL ontologies. In *European Semantic Web Conference*, pages 170–184, 2006.

[105] D. J. Kavvadias and C. H. Papadimitriou. A linear programming approach to reasoning about probabilities. *Annals of Mathematics and Artificial Intelligence*, 1:189–205, 1990.

[106] Y. Kazakov. $\mathcal{SRIQ}$ and $\mathcal{SROIQ}$ are harder than $\mathcal{SHOIQ}$. In *International Conference on Principles of Knowledge Representation and Reasoning*, pages 274–284, 2008.

[107] J. Key, S. Hodgson, R. Z. Omar, T. K. Jensen, S. G. Thompson, A. R. Boobis, D. S. Davies, and P. Elliott. Meta-analysis of studies of alcohol and breast cancer with consideration of the methodological issues. *Cancer Causes Control*, 17:759–770, 2006.

[108] J. M. Keynes. *A Treatise on Probability*. Macmillan, London, 1921.

[109] R. Kindermann and J. L. Snell. *Markov Random Fields and Their Applications*. American Mathematical Society, 1980.

[110] P. Klinov and B. Parsia. Probabilistic modeling and OWL: A user oriented introduction into P-$\mathcal{SHIQ}$(D). In *OWL: Experiences and Directions*, 2008.

[111] P. Klinov and B. Parsia. On improving the scalability of checking satisfiability in probabilistic description logics. In *International Conference on Scalable Uncertainty Management*, volume 5785/2009 of *Lecture Notes in Computer Science*, pages 138–149. Springer Berlin / Heidelberg, 2009.

[112] P. Klinov and B. Parsia. Pronto: A practical probabilistic description logic reasoner. In *International Workshop on Uncertainty in Description Logics*, 2010.

[113] P. Klinov and B. Parsia. Relationships between probabilistic description and first-order logics. In *International Workshop on Uncertainty in Description Logics*, 2010.

[114] P. Klinov, B. Parsia, and D. Picado-Muiño. The consistency of the CADIAG-2 knowledge base: A probabilistic approach. In *International Conference on Logic for Programming, Artificial Intelligence and Reasoning*, pages 432–446, 2010.

[115] P. Klinov, B. Parsia, and D. Picado-Muiño. The consistency of the medical expert system CADIAG-2: A probabilistic approach. *Journal of Information Technology Research*, 4(1):1–20, January 2011.

[116] P. Klinov, B. Parsia, and U. Sattler. On correspondences between probabilistic first-order and description logics. In *International Workshop on Description Logic*, 2009.

[117] G. Klir and T. Folger. *Fuzzy Sets, Uncertainty and Information*. Prentice-Hall International, 1988.

[118] D. Koller and J. Y. Halpern. Irrelevance and conditioning in first-order probabilistic logic. In *Advances in Artificial Intelligence Conference*, pages 569–576, 1996.

[119] D. Koller, A. Levy, and A. Pfeffer. P-CLASSIC: A tractable probabilistic description logic. In *Advances in Artificial Intelligence Conference*, pages 390–397, 1997.

[120] D. Koller and A. Pfeffer. Object-oriented Bayesian networks. In *International Conference on Uncertainty in Artificial Intelligence*, pages 302–313, 1997.

[121] D. Koller and A. Pfeffer. Probabilistic frame-based systems. In *Advances in Artificial Intelligence Conference*, pages 580–587, 1998.

[122] S. Kraus, D. Lehmann, and M. Magidor. Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence*, 44(1–2):167–207, 1990.

[123] H. E. Kyburg. The reference class. *Philosophy of Science*, 50(3):374–397, 1971.

[124] S. C. Lam, J. Z. Pan, D. H. Sleeman, and W. W. Vasconcelos. A fine-grained approach to resolving unsatisfiable ontologies. In *Journal of Web Intelligence*, pages 428–434, 2006.

[125] K. B. Laskey. MEBN: A language for first-order Bayesian knowledge bases. *Artificial Intelligence*, 172(2–3):140–178, 2008.

[126] K. B. Laskey and P. C. G. da Costa. Of starships and *klingons*: Bayesian logic for the 23rd century. In *International Conference on Uncertainty in Artificial Intelligence*, pages 346–353, 2005.

[127] D. Lehmann. Another perspective on default reasoning. *Annals of Mathematics and Artificial Intelligence*, 15(1):61–82, 1995.

[128] H. Leitich, K. Adlassnig, and G. Kolarz. Evaluation of two different models of semiautomatic knowledge acquisition for the medical consultant system CADIAG-2/RHEUMA. *Artificial Intelligence in Medicine*, 25:215–225, 2002.

[129] J. Q. Lew, N. D. Freedman, M. F. Leitzmann, L. A. Brinton, R. N. Hoover, A. R. Hollenbeck, A. Schatzkin, and Y. Park. Alcohol and risk of breast cancer by histologic type and hormone receptor status in postmenopausal women the nih-aarp diet and health study. *American Journal of Epidemiology*, 170(3):308–317, 2009.

[130] L. Lin and Y. Jiang. The computation of hitting sets: Review and new algorithms. *Information Processing Letters*, 86(4):177–184, 2003.

[131] T. Lukasiewicz. Local probabilistic deduction from taxonomic and probabilistic knowledge-bases over conjunctive events. *International Journal of Approximate Reasoning*, 21(1):23–61, 1999.

[132] T. Lukasiewicz. Probabilistic logic programming with conditional constraints. *ACM Transactions on Computational Logic*, 2(3):289–339, 2001.

[133] T. Lukasiewicz. Probabilistic default reasoning with conditional constraints. *Annals of Mathematics and Artificial Intelligence*, 34(1-3):35–88, 2002.

[134] T. Lukasiewicz. Nonmonotonic probabilistic logics under variable-strength inheritance with overriding: Algorithms and implementation in NMPROBLOG. In *International Symposium on Imprecise Probabilities and Their Applications*, pages 230–239, 2005.

[135] T. Lukasiewicz. Nonmonotonic probabilistic logics under variable-strength inheritance with overriding: Complexity, algorithms, and implementation. *International Journal of Approximate Reasoning*, 44(3):301–321, 2007.

[136] T. Lukasiewicz. Expressive probabilistic description logics. *Artificial Intelligence*, 172(6-7):852–883, 2008.

[137] T. Lukasiewicz and U. Straccia. Description logic programs under probabilistic uncertainty and fuzzy vagueness. *International Journal of Approximate Reasoning*, 50(6):837–853, 2009.

[138] C. Lutz, U. Sattler, and L. Tendera. The complexity of finite model reasoning in description logics. *Information and Computation*, 199(1–2):132–171, 2005.

[139] C. Lutz and L. Schröder. Probabilistic description logics for subjective uncertainty. In *International Conference on the Principles of Knowledge Representation and Reasoning*, 2010.

[140] I. Lynce and J. P. M. Silva. On computing minimum unsatisfiable cores. In *International Conference on Theory and Applications of Satisfiability Testing*, 2004.

[141] L. Ma, Y. Yang, Z. Qiu, G. T. Xie, Y. Pan, and S. Liu. Towards a complete OWL ontology benchmark. In *European Semantic Web Conference*, pages 125–139, 2006.

[142] R. M. MacGregor. Inside the LOOM description classifier. *SIGART Bulletin*, 2(3):88–92, 1991.

[143] S. M. Mahoney and K. B. Laskey. Constructing situation specific belief networks. In *International Conference on Uncertainty in Artificial Intelligence*, pages 370–379, 1998.

[144] R. J. McEliece, D. J. C. MacKay, and J.-F. Cheng. Turbo decoding as an instance of *pearl's* "belief propagation" algorithm. *IEEE Journal on Selected Areas in Communications*, 16(2):140–152, 1998.

[145] C. Meilicke and H. Stuckenschmidt. Incoherence as a basis for measuring the quality of ontology mappings. In *Ontology Matching Workshop*, 2008.

[146] C. Meilicke and H. Stuckenschmidt. An efficient method for computing alignment diagnoses. In *Web Reasoning and Rule Systems*, pages 182–196, 2009.

[147] B. Milch and S. J. Russell. First-order probabilistic languages: Into the unknown. In *International Conference on Inductive Logic Programming*, pages 10–24, 2006.

[148] M. Minsky. *The Psychology of Computer Vision*, chapter A Framework for Representing Knowledge. McGraw-Hill, New York, 1975.

[149] P. Mitra, N. F. Noy, and A. R. Jaiswal. OMEN: A probabilistic ontology mapping tool. In *International Semantic Web Conference*, pages 537–547, 2005.

[150] A. M. Mood. *Introduction to the Theory of Statistics*. McGraw-Hill, 1950.

[151] T. H. Näth and R. Möller. ContraBovemRufum: A system for probabilistic lexicographic entailment. In *International Workshop on Description Logic*, 2008.

[152] B. Nebel. Terminological reasoning is inherently intractable. *Artificial Intelligence*, 43(2):235–249, 1990.

[153] R. T. Ng and V. S. Subrahmanian. Probabilistic logic programming. *Information and Computation*, 101(2):150–201, 1992.

[154] G.-B. D. Nguetsé, P. Hansen, and B. Jaumard. Probabilistic satisfiability and decomposition. In *European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, pages 151–161, 1995.

[155] M. Niepert, C. Meilicke, and H. Stuckenschmidt. A probabilistic-logical framework for ontology matching. In *Advances in Artificial Intelligence Conference*, pages 170–184, 2010.

[156] N. J. Nilsson. Probabilistic logic. *Artificial Intelligence*, 28(1):71–87, 1986.

[157] N. J. Nilsson. Probabilistic logic revisited. *Artificial Intelligence*, 59(1–2):39–42, 1993.

[158] Z. Ognjanovic, U. Midic, and J. Kratica. A genetic algorithm for probabilistic SAT problem. In *International Conference on Artificial Intelligence and Soft Computing*, pages 462–467, 2004.

[159] Z. Ognjanovic, U. Midic, and N. Mladenovic. A hybrid genetic and variable neighborhood descent for probabilistic SAT problem. In *Hybrid Metaheuristics*, pages 42–53, 2005.

[160] J. B. Paris and A. Vencovská. In defense of the maximum entropy inference process. *International Journal of Approximate Reasoning*, 17(1):77–103, 1997.

[161] M. Parker and J. Ryan. Finding the minimum weight IIS cover of an infeasible system of linear inequalities. *Annals of Mathematics and Artificial Intelligence*, 17(1–2):107–126, 1996.

[162] M. A. Paskin. Maximum entropy probabilistic logic. Technical Report UCB/CSD-01-1161, University of California, Berkeley, 2002.

[163] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, 2nd printing ed.* Morgan Kaufmann, 1988.

[164] D. Picado Muiño. The (probabilistic) logical content of Cadiag2. In *Proceedings of ICAART 2010*, pages 28–35, 2010.

[165] J. L. Pollock. *Nomic Probabilities and the Foundations of Induction.* Oxford University Press, 1990.

[166] L. Predoiu and H. Stuckenschmidt. Probabilistic extensions of semantic web languages: A survey. In *The Semantic Web for Knowledge and Data Management: Technologies and Practices.* Idea Group Inc, 2009.

[167] H. Reichenbach. *The Theory of Probability.* University of California Press, 1949.

[168] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.

[169] R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32:57–95, 1987.

[170] M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62(1–2):107–136, 2006.

[171] N. Robertson and P. D. Seymour. Graph minors *iii*: Planar tree-width. *Journal of Combinatorial Theory, Series B*, 36(1):49–64, 1984.

[172] Rock — online breast cancer knowledge base. http://rock.icr.ac.uk/, September 2010. Breakthrough Toby Robins Breast Cancer Research Centre at the Institute of Cancer Research.

[173] U. Sattler, T. Schneider, and M. Zakharyaschev. Which kind of module should I extract? In *International Workshop on Description Logic*, 2009.

[174] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.

[175] T. A. Sellers, R. A. Vierkant, J. R. Cerhan, S. M. Gapstur, C. M. Vachon, J. E. Olson, V. S. Pankratz, and L. H. Kushi. Interaction of dietary folate intake, alcohol, and risk of hormone receptor-defined breast cancer in a prospective study of postmenopausal women. *Cancer Epidemiology, Biomarkers and Prevention*, 11:1104–1107, 2002.

[176] R. Shearer, B. Motik, and I. Horrocks. HermiT: A highly-efficient OWL reasoner. In *OWL: Experience and Directions*, 2008.

[177] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical OWL-DL reasoner. *Journal of Web Semantics*, 5(2):51–53, 2007.

[178] A. C. Society. What are the risk factors for breast cancer? http://www.cancer.org/Cancer/BreastCancer/DetailedGuide/breast-cancer-risk-factors, September 2010.

[179] G. Stoilos, G. B. Stamou, J. Z. Pan, V. Tzouvaras, and I. Horrocks. Reasoning with very expressive fuzzy description logics. *Journal Artificial Intelligence Research*, 30:273–320, 2007.

[180] U. Straccia. Reasoning within fuzzy description logics. *Journal of Artificial Intelligence Research*, 14:137–166, 2001.

[181] R. Suzuki, W. Ye, T. Rylander-Rudqvist, S. Saji, G. A. Colditz, and A. Wolk. Alcohol and postmenopausal breast cancer risk defined by estrogen and progesterone receptor status: A prospective cohort study. *Journal of the National Cancer Institute*, 97(21):1601–1608, 2005.

[182] D. Tsarkov and I. Horrocks. FaCT++ description logic reasoner: System description. In *International Joint Conference on Automated Reasoning*, pages 292–297, 2006.

[183] D. Tsarkov, I. Horrocks, and P. F. Patel-Schneider. Optimizing terminological reasoning for expressive description logics. *Journal of Automated Reasoning*, 39(3):277–316, 2007.

[184] L. C. van der Gaag. Computing probability intervals under independency constraints. In *International Conference on Uncertainty in Artificial Intelligence*, pages 457–466, 1990.

[185] C. J. van Rijsbergen. *Information Retrieval*. Butterworth, 1979.

[186] J. von Kries. *Die Principien der Wahrscheinlichkeitsrechnung und Rational Expectation*. Freiburg, 1886.

[187] P. Walley, R. Pelessoni, and P. Vicig. Direct algorithms for checking consistency and making inferences from conditional probability assessments. *Journal of Statistical Planning and Inference*, 126(1):119–151, 2004.

[188] M. P. Wellman and J. S. Breese. From knowledge bases to decision models. *Knowledge Engineering Review*, 7(1):35–53, 1992.

[189] J. Williamson. Objective Bayesian probabilistic logic. *Journal of Algorithms in Cognition, Informatics and Logics*, 63(4):167–183, 2008.

[190] H.-T. Zheng, B.-Y. Kang, and H.-G. Kim. An ontology-based Bayesian network approach for representing uncertainty in clinical practice guidelines. In *Uncertainty Reasoning in the Semantic Web Workshop*, pages 161–173, 2008.

[191] H. Zimmermann. *Fuzzy Set Theory and its Applications*. Kluwer Academic Publisher, Boston - Dordrecht - London, 1991.