Hybrid Probabilistic Logics
Theoretical Aspects, Algorithms and Experiments

Proefschrift

Ter verkrijging van de graad van doctor
aan de Radboud Universiteit Nijmegen
op gezag van de rector magnificus,
volgens besluit van het college van decanen
in het openbaar te verdedigen op woensdag 4 mei 2016
om 14:30 uur precies
door

Steffen Michels

geboren 3 juni 1985 te Kleve, Duitsland

SIKS Dissertation Series No. 2016-15
The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems.

This thesis was supported by the Dutch national program COMMIT. The research work was carried out as part of the Metis project under the responsibility of Embedded Systems Innovation by TNO, with Thales Nederland B.V. as the carrying industrial partner.

Cover head gears: Lightspring/Shutterstock
Cover drawings of ships: Anna Berkemeyer
Printed by: Gildeprint
ISBN: 978-94-6233-253-9

CONTENTS

1 INTRODUCTION

1.1 DECISION MAKING AND UNCERTAINTY

Decision making for specific situations by humans is usually done in the face of limited availability of knowledge about these situations as well as about general principles according to which the world behaves. Developing an understanding of such reasoning is not only crucial for decision making, but also as a foundation for empirical science. Moreover, without such an understanding one cannot build intelligent systems that are able to operate in an uncertain world. For these systems, it is essential to have the ability to distinguish between valid and invalid conclusions about the world and a way to compare the effects of decisions and actions, even if these effects are uncertain.

A well-accepted theory, which lays the foundation for such understanding, is *probability theory*. It is based on the idea of measuring uncertainty in terms of probabilities, i.e. real numbers in the range between 0 and 1, with a foundation based on mathematical measure theory. Probability theory provides operations to combine uncertain *events* and ways to update probabilities by incorporating *evidence*. However, this is only the first step, as also needed are methods that allow humans to express uncertain knowledge and support uncertainty reasoning in a structured and efficient manner.

1.2 FROM LOGICAL TO PROBABILISTIC REASONING

Before we come to probabilistic reasoning, we start with a discussion of *logical* reasoning, as this is well understood and provides important insights for probabilistic reasoning.

Logic already emerged in ancient times and is not only concerned with an abstract mathematical structure of logical reasoning, but also offers a language to express knowledge in a structured way, conceivable by humans. This structured way of expressing knowledge is also the basis for reasoning, by employing reasoning steps that agree with human intuition. For instance, Aristotle proposed a certain structure of how knowledge should be represented and came up with general principles of how to reason with such structured knowledge, which is nowadays referred to as *syllogistic*. This paved the way for the development of modern formal logics.

Modern formal logics not only provide languages for expressing knowledge in a structured, unambiguous way, but also provide rigid, mathematical definitions of what distinguishes valid from invalid conclusions. To mechanically draw conclusions, logics come with *inference* rules which can be applied to any knowledge represented in logic, thereby abstracting from the actual knowledge expressed. An important property of such inference methods is that they provide only valid conclusions, based on the mathematics of validity. Such inference systems are said to be *sound*. The consequence of this is that wrong conclusions, in the sense of being different from what is expected in the real world, are always a *consequence* of the knowledge used and never due to reasoning steps. It is acceptable that an inference procedure is not able to draw conclusions in all cases, i.e. it is not necessarily *complete*, as this cannot lead to incorrect conclusions. Typically, the reason for an inference method to be incomplete is that it is either computationally too expensive or even impossible to provide all valid conclusions for a certain logic.

This complexity of reasoning is highly related to the expressivity of a language: there is always a trade-off between expressivity, on the one hand, and complexity and decidability of reasoning, on the other hand [86]. For logic, there are different ways of structuring knowledge with varying expressivity. An important aspect is the ability to express general knowledge for a group of objects, which is crucial to make the representation of general knowledge possible. Such abstraction is provided by *variables* and *quantifiers* in *first-order logic* (FOL). There are more expressive logics, such as *higher-order logic* (HOL), which allows one to express general knowledge not only about objects, but also about properties of objects. This increased expressivity has significant influence on the complexity and decidability of reasoning.

Probability theory, on the other hand, offers mathematical definitions of (conditional) probabilities of events, but it does not provide means to define *probability measures* and ways for structural reasoning with them. For probabilistic reasoning, it is however crucial to have reasoning methods with similar properties as for logical reasoning. A way to achieve this is to combine logics with probability theory. Such combinations are referred to as *probabilistic logics*. Similar to logical reasoning, a significant challenge is to take into account the trade-off between expressivity and the complexity of reasoning.

There are two fundamentally different ways of combining logical and probabilistic reasoning. At first sight, probabilistic reasoning can be seen as a special case of logical reasoning. As logical reasoning is very general, one can use logic to formalise probability theory and reason about probability measures. Representative for this approach is the work on probabilistic logics by Bacchus [6] and Halpern [63]. On the other hand, logical reasoning can be viewed as a special case of probabilistic reasoning, with deterministic probabilities, 0 and 1, only. We consider dealing with uncertainty as a central issue, which is why we believe that probabilistic reasoning should be seen as a generalisation of logical reasoning. Such reasoning yields, instead of deterministic conclusions, probability distributions over conclusions. This point of view is also the basis for most practical probabilistic logic languages in the context of *artificial intelligence* (AI), emerging from the early work of Nilsson [113].

Concretely, probabilistic logics use logics to express probabilistic events and knowledge about their probabilities in a structured way. As for logical reasoning, we want to provide sound inference methods, meaning that the conclusions drawn about the events' probabilities are always correct. We can use different logics as a basis for probabilistic logics, which as for standard logics provide a different balance between expressivity, and complexity and decidability of inference.

1.3 MAKING PROBABILISTIC LOGICAL LANGUAGES FEASIBLE

The field of AI aims at automating reasoning for realistic problems. To achieve this, languages are employed that must not only be expressive enough, but should also allow to structure the knowledge in a way suited for the application domain at hand. Similarly, reasoning must not only have desirable theoretical complexity properties, but one requires efficient implementations of inference algorithms, such that conclusions can be computed within a specific time, acceptable for the application.

The first successful languages for structured probabilistic knowledge were based on graphs. *Bayesian networks* (BNs) [119] are a prominent example of such a language. These languages have, despite their limited expressivity, successfully been employed for serious, real-world problems, as the probabilistic aspect of a problem is implicitly included in the models and does not have to be encoded graphically. However, in the same way as for logics, abstract knowledge, in particular knowledge abstracting from concrete objects, so being at least first-order, requires textual languages. Probabilistic logic languages employ variables to provide the ability to express first-order knowledge, similarly as ordinary logical languages.

As for the various non-probabilistic logic languages, different probabilistic logic languages provide different balances between expressiveness, complexity and decidability of reasoning. On one side of the spectrum of probabilistic logics, there are languages enforcing little structure on how probabilistic information is specified. As a consequence, probability measures can be underspecified, i.e. the probability of statements is only bounded and not uniquely determined, and definitions can even become inconsistent. Such underspecified probabilities are also called *imprecise* probabilities. This is analogous to classical logics, in which a theory may not be able to determine whether a statement is true or false and where theories can become inconsistent. The work on such logics (e.g. Nilsson [113]) is moreover more of a theoretical nature and does not provide efficient algorithms for reasoning.

On the other end of the spectrum of probabilistic logics, languages have been proposed that enforce a structured definition of a unique probability measure. The structure provided by such languages can be exploited for efficient inference and indeed such languages have successfully been employed for solving practical problems. There are numerous languages and implementations available, ranging from *Markov logic networks* (MLNs) [130], which employ weights instead of probabilities to ensure unique distributions, to a family of languages, based on *logic programming* (LP), which provides similar structure as BNs, for instance *independent choice logic* (ICL) [122], *PRISM* [138], *ProbLog* [125] and *causal probabilistic logic* (CP-logic) [152].

However, enforcing a unique probability distribution has major drawbacks. First, it requires the modeller to be more precise than might be justified by the knowledge available, which may lead to decisions that are not justified. Second, it causes many issues when dealing with an infinite number of states, which is already the case as soon as a single real number or integer without bounded range is used to model the world. The ability to deal with numbers is certainly crucial for most realistic problems. However, as computing exact probabilities requires the consideration of an infinite number of states, such computations become incomputable in general. The only way to still allow for sound inference, is to severely restrict the expressiveness of a language, i.e. to restrict the distributions used and how they can be combined. This dramatically hinders the applicability of such languages to many real-world problems.

One possible solution for the latter problem is to compute approximations of probabilities instead of computing exact ones. There are numerous algorithms, mainly based on sampling, that can compute good approximations even for incomputable problems. A limitation of such inference methods is that they do not provide general guarantees about the quality of the result. This can again lead to wrong decisions.

Languages supporting imprecise probabilities can handle both issues mentioned above. Firstly, imprecise probabilities overcome the issue that one is forced to specify knowledge more precisely than is actually justified. In contrast to specifying precise probabilities they allow one to only constrain the probabilities of events as much as justified by the knowledge available. A consequence of this is that one may not be able to determine what the best decision is. So imprecise reasoning is cautious in the sense that it prefers a situation in which one cannot decide what the best decision is, above a situation in which a wrong conclusion seems as a correct one. The second advantage of imprecise reasoning is that it can be made sound for probabilistic reasoning over infinite spaces. An underspecified set of probability distributions over an infinite space can still be represented by a finite number of probabilistic constraints. As an example, suppose that the probability of the temperature being between 19 °C and 21 °C is 0.9. One can *soundly* conclude from this that the probability of a temperature above 15 °C is at least 0.9, but has to accept the imprecision that it can be higher as well. This information may not be enough to determine what the best decision is, but in case it is, this conclusion is sound.

There are however some drawbacks of using languages that support imprecise probabilities, as they usually impose little structure. A consequence of this is that knowledge can quickly become inconsistent and this is hard to solve for a user of the language, as it might be the result of complex interactions between pieces of knowledge. The lack of structure also hinders efficient inference, as exploiting the structure of problems is the key for making inference efficient. Even for imprecise languages which provide much structure, such as extensions of BNs (e.g. *locally defined credal networks* (LDCNs) [31]), inference is strictly more complex than for the precise case.

1.4 CONTRIBUTIONS OF THIS THESIS

This thesis argues in favour of imprecise probabilistic logic reasoning, while preserving important properties by the provision of enough structure. An important aim is to keep a balance between the two extremes of the spectrum of languages mentioned above. The main guarantee we strive for, is to prevent the occurrence of inconsistencies, unlike previous imprecise probabilistic logics. Also we show that we can achieve imprecise inference complexity, similar to precise inference complexity.

To demonstrate the practical applicability of the underlying ideas, it is not enough to only provide a general reasoning framework, but also one has to provide actual languages that offer support for the natural representation of knowledge. So in this thesis we provide two languages with different aims in terms of which kind of knowledge can be expressed and with different guarantees. The design of a language is closely connected to considerations about what sort of knowledge one wishes to represent and the reasoning capabilities that should be supported. Since these considerations range from theoretical to practical ones, a similar range of results, from the theoretical underpinning of the languages to its actual implementation, can be found in this thesis. The results support sound probabilistic reasoning with models having infinitely many states and for cases where probabilistic information is missing.

We also show applicability of our results for a concrete problem domain. We built a model for maritime safety and security tasks, using one of the languages proposed. This probabilistic logic is very suited for the domain for a number of reasons. First, knowledge representation is essential in this domain, as labelled data about safety and security tasks is hardly available. This makes methods solely based on machine learning unsuitable. Second, rational reasoning involving numeric information with infinitely many states is essential, as numeric aspects are very important in this domain. Lastly, since actual decisions in the domain can have a huge impact, it is important that guarantees about probabilities on which the decisions are based can be given.

1.5 OUTLINE OF THE THESIS

We give an overview of the thesis' chapters and the publications on which they are based.

Chapter 2: PRELIMINARIES

In this chapter we give a brief overview of the principles of the two basic theories the work of this thesis is based on: logic and probability theory.

Chapter 3: PROBABILISTIC LOGICS

This chapter offers an overview of work in which logics and probability theory are combined and also compares several probabilistic logics. The chapter focuses more on a conceptual level on properties of families of languages, instead of concrete implementation choices, with the goal of providing the necessary context for the work in this thesis. The content of this chapter is so far unpublished. My contribution consisted of writing this overview.

Chapter 4: A NEW PROBABILISTIC CONSTRAINT LOGIC

This chapter introduces a novel probabilistic logic: *Probabilistic Constraint Logic Programming* (PCLP). PCLP is more expressive than many modern languages, which mainly aim at efficient inference, as it supports arbitrary *random variable* ranges while preserving computability of reasoning by utilising underspecified probability distributions, i.e. imprecise probabilities. At the same time, it has similar characteristics as many other modern probabilistic logics in terms of consistency guarantees. This chapter is based on Sections 2, 4–6 and 8.1 of the following article:

"A new probabilistic constraint logic programming language based on a generalised distribution semantics" [99]

I contributed the design of the language as well as the theoretical results, concretely the definition of the semantics and the proofs of its properties. I also did most of the writing.

The principles the work is based upon were already published before in:

"Inference for a New Probabilistic Constraint Logic" [96]

I provided the basic idea the work is based on, contributed to the theoretical development and the writing, and designed and implemented the inference algorithm and carried out the experiments.

Chapter 5: INFERENCE BY GENERALISED WEIGHTED MODEL COUNTING

In this chapter we introduce an inference algorithm for PCLP, which is a generalisation of *weighted model counting* (WMC) algorithms, typically employed for inference for commonly applied probabilistic logics, defining precise distributions. The results have a theoretical and a practical nature. Theoretically, we show that inference for PCLP has complexity characteristics, comparable to the above mentioned precise probabilistic logics, which is exceptional for an imprecise language. Practically, we show as well that the algorithm can exploit the structure of problems, similarly to how this is possible for precise problems. This illustrates the practical applicability of PCLP, which is further supported by experiments. The basis for this chapter are Sections 7 and 8.2 of:

"A new probabilistic constraint logic programming language based on a generalised distribution semantic" [99]

I designed and implemented the algorithm and performed the experiments. Also the main part of the theoretical work and the writing are my contribution.

An early version of the inference algorithm was already included in the paper:

"Inference for a New Probabilistic Constraint Logic" [96]

Chapter 6: EFFECTIVE APPROXIMATION OF HYBRID DISTRIBUTIONS WITH BOUNDED ERROR

In this chapter we further illustrate the practical applicability of the inference principle developed in Chapter 5 for approximating hybrid distributions with bounded error. This is done by providing an algorithm that iteratively refines imprecise approximations of precise, hybrid distributions in an effective way. The effectiveness is shown by experimental comparison with state-of-the-art sampling-based inference algorithms. The chapter is based on the paper:

"Approximate Probabilistic Inference with Bounded Error for Hybrid Probabilistic Logic Programming" [100]

My contribution includes the design and implementation of the algorithm, carrying out the experiments and the main part of the writing.

Chapter 7: IMPRECISE PROBABILISTIC HORN CLAUSE LOGIC

The chapter introduces another probabilistic logic, which allows for imprecise reasoning: *imprecise probabilistic Horn clause logic* (IPHL). This logic is geared towards imprecise knowledge representation for discrete problems and has the unique property that imprecise reasoning is equally expensive as inference for corresponding precise problems. The basis for this chapter is the paper:

"Imprecise Probabilistic Horn Clause Logic" [98]

My contribution consisted of collaborating in designing the language, providing the inference method with proofs of correctness and complexity properties and main parts of the writing.

Chapter 8: APPLICATION TO MARITIME SAFETY AND SECURITY TASKS

In this chapter we show an application of PCLP in the context of the maritime safety and security domain. We discuss the principles of a model for reasoning about situations, based on the uncertain information available in this domain, with a focus on illustrating applicability of PCLP. Effectiveness of the proposed model is shown qualitatively and quantitatively. Additionally, real-world applicability is illustrated by discussing the successful integration in a complex prototype system, developed in the course of the *METIS* project. The chapter is based on:

"A Decision Support Model for Uncertainty Reasoning in Safety and Security Tasks" [97]

I developed and implemented the model, performed the experiments and contributed to the writing.

More details of the model were also given in a technical report:

"A Probabilistic Logic-based Model for Fusing Attribute Information of Objects Under Surveillance" [95]

The description of the integration in the METIS prototype in this chapter (Section 8.4.3) is partly based on Section 4 of the paper:

"An Integrated Reconfigurable System for Maritime Situational Awareness" [151]

My contribution was the active participation in the METIS project, which made possible the development of the prototype, that integrates components from our industrial and various academic partners.

Chapter 9: DISCUSSION AND CONCLUSIONS

In this chapter we look back on what we have achieved in the research described in this thesis and offer an outlook on directions for future work.

2 PRELIMINARIES

In this chapter we introduce the two main concepts the work of the thesis is based on: *logic* (Section 2.1) and *probability theory* (Section 2.2).

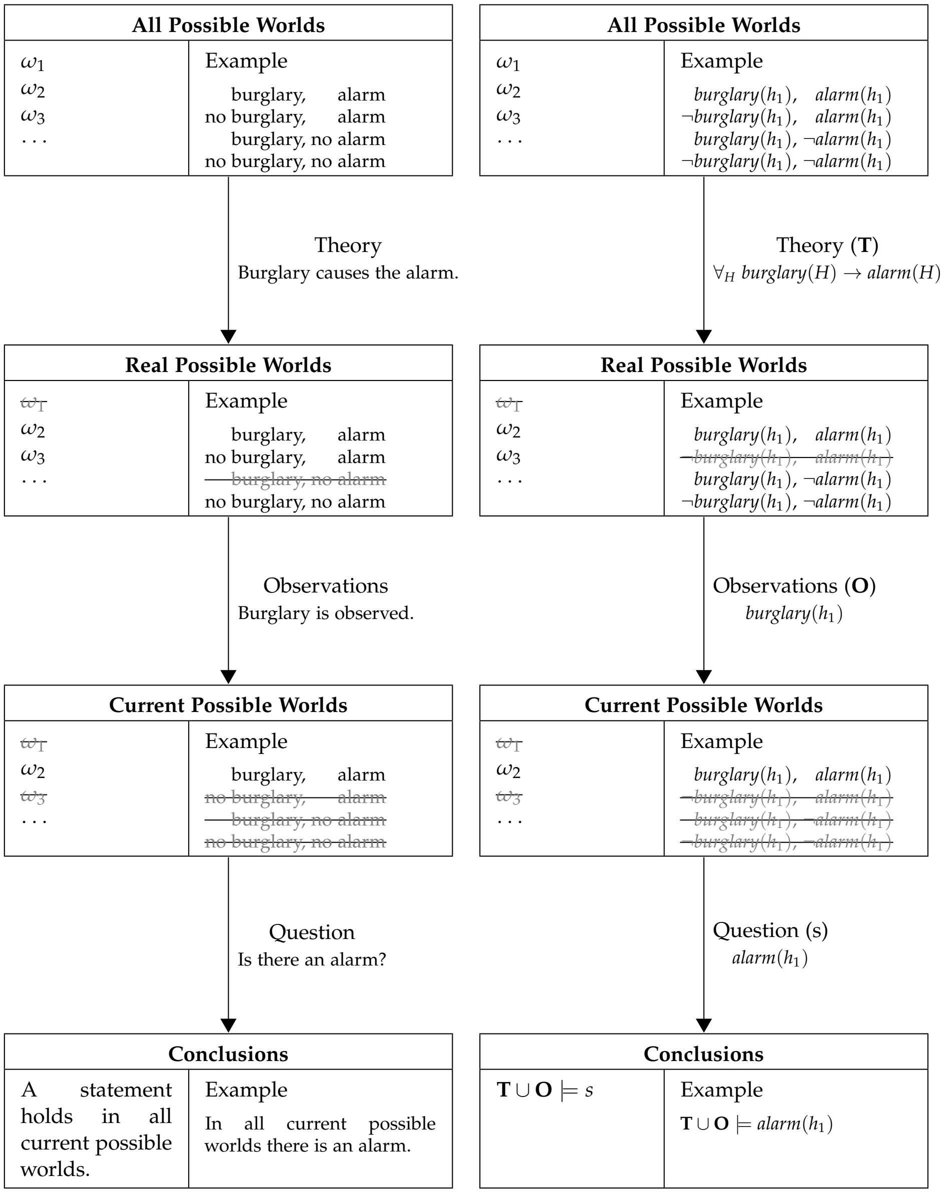
2.1 LOGIC

In this section we introduce the concepts behind modern formal logic. The logics we discuss in particular in this section are *first-order logic* (FOL) and *logic programming* (LP), which are significantly important for *artificial intelligence* (AI). Modern formal logics emerged from the work of Boole in the eighteenths century, who first put logic in a mathematical, in particular an algebraic context, meaning that logical reasoning can be viewed as computations [14]. Frege laid the basis for a formal language expressing assertions about the world, particularly the use of quantifiers to express general knowledge [50]. The development of modern mathematical logic laying the foundation for the languages treated here, is based largely on the work of Gödel and Tarski in the 1930s.

2.1.1 *Possible Worlds*

The basic concept underlying logics is that of *possible worlds*. Possible worlds are in logic also called *structures* or *models*. We however use the term possible world, which generalises the concept as we also apply it to probability theory, which we introduce later. Logical reasoning then means starting from all possible states of the world and using statements about the world to restrict the set of possible worlds, thereby increasing the knowledge about the world. The basic idea is illustrated in Figure 2.1a.

Once the set of possible worlds is defined, knowledge about the world is expressed by a logical *theory*, also called *knowledge base*. This theory expresses in general which worlds are actually possible in reality and which not. The strength of formal logic is that it allows one to mathematically define which possible worlds are consistent with a theory and which are not. The set of possible worlds is further restricted by observations of the current situation. In some cases no observations are required, e.g. the theory is a set of mathematical axioms and we want to conclude theorems from it. In theories about the real world however there is often the natural distinction between general knowledge about objects


All Possible Worlds
ω_1 ω_2 ω_3 …
Example
burglary, alarm
no burglary, alarm
burglary, no alarm
no burglary, no alarm
Theory
Burglary causes the alarm.
Real Possible Worlds
~~ω_1~~ ω_2 ω_3 …
Example
burglary, alarm
no burglary, alarm
~~burglary, no alarm~~
no burglary, no alarm
Observations
Burglary is observed.
Current Possible Worlds
~~ω_1~~ ω_2 ~~ω_3~~ …
Example
burglary, alarm
~~no burglary, alarm~~
~~burglary, no alarm~~
~~no burglary, no alarm~~
Question
Is there an alarm?
Conclusions
A statement holds in all current possible worlds.
Example
In all current possible worlds there is an alarm.
All Possible Worlds
ω_1 ω_2 ω_3 …
Example
$burglary(h_1),\ alarm(h_1)$
$\neg burglary(h_1),\ alarm(h_1)$
$burglary(h_1),\ \neg alarm(h_1)$
$\neg burglary(h_1),\ \neg alarm(h_1)$
Theory (**T**)
$\forall_H\ burglary(H) \rightarrow alarm(H)$
Real Possible Worlds
~~ω_1~~ ω_2 ω_3 …
Example
$burglary(h_1),\ alarm(h_1)$
~~$\neg burglary(h_1),\ alarm(h_1)$~~
$burglary(h_1),\ \neg alarm(h_1)$
$\neg burglary(h_1),\ \neg alarm(h_1)$
Observations (**O**)
$burglary(h_1)$
Current Possible Worlds
~~ω_1~~ ω_2 ~~ω_3~~ …
Example
$burglary(h_1),\ alarm(h_1)$
~~$\neg burglary(h_1),\ alarm(h_1)$~~
~~$burglary(h_1),\ \neg alarm(h_1)$~~
~~$\neg burglary(h_1),\ \neg alarm(h_1)$~~
Question (s)
$alarm(h_1)$
Conclusions
$\mathbf{T} \cup \mathbf{O} \models s$
Example
$\mathbf{T} \cup \mathbf{O} \models alarm(h_1)$

(a) General Logical Reasoning

(b) FOL Reasoning

Figure 2.1: Logical Reasoning Illustrations

in the world and current observations of specific objects. The distinction between the theory and the observations can be purely conceptual, in which case technically there is no difference and together they just form a single theory, but both can also be treated differently in some formalisms, i.e. observations are treated in a special way. The distinction will especially become important when we move on to probability theory. Finally, conclusions are drawn based on the remaining possible worlds. Concretely, it is usually concluded that some statement holds in all possible worlds or not.

Example 2.1

We use a classic example commonly used to illustrate uncertainty reasoning as a running example to illustrate the properties of logical, probabilistic and combined formalisms. Suppose we want to model a house with an alarm system detecting burglars and just model the possible states of the world as the possible combinations of statements whether there is a burglar and an alarm or not, as shown in Figure 2.1a. The general theory states that burglary causes the alarm, which means that the possible world in which there is a burglar, but no alarm, is inconsistent with the theory. In case we observe that there is a burglar we can also exclude two more possible worlds. We can conclude that there is an alarm, as this holds in the only possible world remaining.

2.1.2 *First-Order Logic*

FOL is a widely applied *predicate* logic, where predicates are used to model relations between specific objects and *quantified variables* are used to express general relations abstracting from specific objects. This is in contrast to so-called *propositional* logic, which only allows one to express statements and does not explicitly distinguish between objects and relationships. The abstraction offered by quantified variables is the key property to make FOL suited for developing real-world models. It allows one to express complex relations between a large number of objects, but still keep the complexity of reasoning and learning manageable. A classic example to illustrate how FOL compares to propositional logic, is that the rules of chess can be encoded in FOL about 100 000 times more compactly than with a propositional theory. The natural statement "A pawn can move to an unoccupied square in front." can be translated to FOL quite directly. In contrast, a propositional theory would have to include statements such as "The first white pawn can move from a3 to a4 in case the last black pawn is not on a4 and the second white pawn is not on a4 and . . . ".

An important property of FOL is that it is *complete*, as proven by *Gödel's completeness theorem* [55]. That means semantic truth and syntactic provability correspond to each other; so if something is proven it is also semantically true and all semantically true statements can also be proven. However, FOL is restricted in its expressiveness, as it only allows to quantify over objects and not over predicates. For example, one can express that in chess all rooks have certain properties, but not that all properties possessed by rooks also transfer to the queen. This is allowed in *second-order logic* which comes with the price that important properties, such as completeness, do not hold any more. We therefore restrict to FOL, which is expressive enough to model a wide range of AI problems. FOL is still only semi-decidable, which can hardly be avoided for a sufficiently expressive knowledge representation formalism.

2.1.2.1 *Syntax & Semantics*

As discussed, FOL is concerned with relations between objects in the world. The objects in the world are formalised as the *domain of discourse* $\mathbf{D}$, which can be an arbitrary set. The bridge from the objects in the domain of discourse and the logical language is formed by *terms*, composed of functions. In the language of FOL such functions are represented by function symbols with associated arity; functions of arity 0 are also called *constants*. A function symbol of arity n is interpreted as function from $\mathbf{D}^n$ to $\mathbf{D}$ in a possible world. Predicates are used to represent relations between objects. For which objects the predicates hold is also fixed in a possible world. We denote function symbols starting with lower case letters (e.g. $person_1$, $motherOf, \ldots$). Predicate symbols also have an arity and are interpreted as functions from $\mathbf{D}^n$ to $\{true, false\}$, indicating whether the relation holds between the objects given or not. We denote predicates also with lower case letters (e.g. $alarm$, $motherOf, \ldots$). Note that $motherOf$ could be meaningful as a 1-ary function, e.g. $motherOf(person_1)$ could represent the mother of $person_1$, or as 2-ary predicate, e.g. $motherOf(person_1, person_2)$ could mean $person_1$ is the mother of $person_2$.

A possible world is then formalised as a domain of discourse $\mathbf{D}$ and a so-called *interpretation* $\mathcal{I}$, determining the functions and relations:

- For each n-ary function symbol f, $\mathcal{I}(f)$ is a function from $\mathbf{D}^n$ to $\mathbf{D}$.
- For each n-ary predicate symbol p, $\mathcal{I}(p)$ is a function from $\mathbf{D}^n$ to $\{true, false\}$.

Example 2.2

We continue with Example 2.1, but this time consider multiple houses instead of a single one. In case there are n houses, the domain of discourse could for instance be:

$$\mathbf{D} = \{house_1, \ldots, house_n\}$$

Here we do not distinguish between burglars, so we can only model that there is a burglar present in a particular house, but cannot say anything more about burglars. With this, we can use the two predicates *burglary* and *alarm* of arity 1 to model whether a burglar is present in a house and the alarm is on. The possible worlds for a single house ($\mathbf{D} = \{h_1\}$) are illustrated in Figure 2.1b. In general, there are 2^{2n} possible worlds for n houses. The notation is chosen for convenience. Formally, $alarm(h_1)$ means $\mathcal{I}(alarm)(h_1) = true$ and $\neg alarm(h_1)$ means $\mathcal{I}(alarm)(h_1) = false$. Also we assume that the function symbols $h_1, \ldots, h_n$ match the objects in the domain of discourse, i.e. that for all i: $\mathcal{I}(h_i) = house_i$.

Logical theories are built by formulas, corresponding to statements about the world. The basic building blocks of logic formulas are predicates applied to terms, which are build from function applications. Examples are *burglary*(h_1) and *rich*$(\mathit{ownerOf}(h_1))$. Formulas can be combined using operators. Some examples are:

- conjunctions $(f \wedge g)$: f and g are both true.
- disjunctions $(f \vee g)$: f or g (or both) are true.
- implications $(f \rightarrow g)$: In case f is true, g is true as well.
- equivalences $(f \leftrightarrow g)$: f and g are either both true or both not true.

The meaning of such operators is usually defined by so-called truth tables, of which examples are given in Table 2.1. Also formulas can be modified by negations $(\neg)$. There is some redundancy in the above set of operators. For instance, $f \leftrightarrow g$ can be expressed as $f \rightarrow g \wedge g \rightarrow f$, which in turn is equivalent to $(\neg f \vee g) \wedge (\neg g \vee f)$. However, the various operators are used for convenience and intuitively match important concepts of rational reasoning.

f	g	$f \wedge g$	$f \vee g$	$f \rightarrow g$	$f \leftrightarrow g$
$\bot$	$\bot$	$\bot$	$\bot$	$\top$	$\top$
$\bot$	$\top$	$\bot$	$\top$	$\top$	$\bot$
$\top$	$\bot$	$\bot$	$\top$	$\bot$	$\bot$
$\top$	$\top$	$\top$	$\top$	$\top$	$\top$

Table 2.1: Truth Table for Logical Connectives

To represent general knowledge abstracting from concrete objects, variables are used. Such variables can be used instead of terms (e.g. *burglary*(H)). We denote variables starting with upper case letters (e.g. *House*). Variables have to be bound by either universal quantification $(\forall)$ or existential quantification $(\exists)$, e.g. $\forall_X\, f(X)$ binds X and means that f holds for all objects in the domain of discourse. Existential quantification means that there is at least one object in the domain of discourse for which the formula holds.

A formula is called a *sentence* in case all variables are bound, as only then it represents a self-contained statement about the world. A logical theory is a set of sentences assumed to always hold. One can conclude that a sentence s is always true given a theory $\mathbf{T}$, in case s is true in all possible worlds allowed by $\mathbf{T}$ and denote this as:

$$\mathbf{T} \models s$$

We also say in this case that the theory is a model of the sentence or the sentence is a *consequence* of the theory. There is also an alternative formulation of that a sentence can be concluded from a theory. The statement $\mathbf{T} \models s$ is equivalent to the statement that the following sentence is *valid*:

$$\left(\bigwedge_{t \in \mathbf{T}} t\right) \rightarrow s$$

Here valid means that it is true in all possible worlds.

This definition of how to draw conclusions can be problematic in case the theory becomes inconsistent, i.e. no possible world is consistent with it. In this case, every possible statement is a consequence of the theory, which is usually unwanted in practice. The situation often occurs when adding observations to a general theory, which overspecifies the domain, meaning that it forbids cases which can actually occur in reality.

The notion of consequence is actually not only restricted to all possible worlds given a single domain of discourse, but applies to all possible domains of discourse. Usually if building a theory one has a specific *intended interpretation* in mind, but all knowledge about the structure of the domain of discourse can in FOL be encoded in the theory, the actual domain of discourse and meaning of functions does not matter for reasoning. In the previous examples the domain of discourse was given only as illustration. As discussed later, in other logics and in case we want to define a probability distribution over possible worlds however it is often desirable to fix the domain of discourse and interpretation of functions, called a *pre-interpretation*.

Given a possible world, whether a sentence is true can be determined formally by the so-called *truth schema* (T-schema) [145]. As we inductively break down the formula, sub-formulas are not necessarily sentences, i.e. they may contain free variables. So additionally to the possible world we require a *valuation* μ, which is a function from all variable symbols to objects in the domain of discourse. The truth value of sentences does not depend on this valuation, so initially we can start with an arbitrary one. Then the T-schema is defined as follows:

1. $\forall_X f$ is true given μ, if for all possible valuations $\mu'_i, \ldots$, differing in μ only on the value of X, f is true.

2. $\exists_X f$ is true given μ, if there is a valuation μ', differing in μ only on the value of X, such that f is true.

3. $\neg f$ is true, if f is not true.

4. $f \odot g$ (where $\odot$ is some operator) is evaluated according to the truth table of $\odot$.

5. $p(t_1, \ldots, t_n)$ is true in case $p(v_1, \ldots, v_n)$ is true given the current interpretation $(\mathcal{I}(p(v_1, \ldots, v_n)) = \mathit{true})$, where $v_1, \ldots, v_n$ are the valuations of $t_1, \ldots, t_n$ according to μ and $\mathcal{I}$.

Example 2.3

How we can formally reason in FOL about the burglary example is shown in Figure 2.1b. That burglary causes an alarm is expressed as $\mathbf{T} = \{\forall_H \mathit{burglary}(H) \rightarrow \mathit{alarm}(H)\}$, which means that for all objects (only houses in this case) in case there is a burglar, there also has to be an alarm. The observation that there is a burglar in a specific house can be expressed by a single sentence, without variables: $\mathbf{O} = \{\mathit{burglary}(h_1)\}$.

We can now conclude that there is an alarm in house h_1 ($\mathbf{T} \cup \mathbf{O} \models \mathit{alarm}(h_1)$). For another house we cannot draw that conclusion ($\mathbf{T} \cup \mathbf{O} \not\models \mathit{alarm}(h_2)$), as we do not know whether there is a burglar in this house or not. If we consider the alternative observation that there is no burglar ($\mathbf{O}' = \{\neg \mathit{burglary}(h_1)\}$), we can also not conclude that there is no alarm ($\mathbf{T} \cup \mathbf{O}' \not\models \neg \mathit{alarm}(h_1)$), as the rule $\forall_H\ \mathit{burglary}(H) \rightarrow \mathit{alarm}(H)$ only tells there is an alarm in case of burglary, but does not say anything about the case there is no burglary. There are still two possible worlds which do not agree whether there is a burglar or not.

2.1.2.2 *Automating Reasoning: Herbrand's Theorem*

The challenge for determining whether a sentence is a consequence of a theory, or equivalently whether a sentence is valid, is that one has to consider all possible worlds of which there are infinitely many. A fundamental result of Herbrand is that only a fixed, finite domain of discourse is necessary to prove that a sentence is valid [67]. We present this result as a *refutation* method, meaning the method tries to derive that the negation of the sentence is *unsatisfiable*, i.e. it is false in all possible worlds, which equivalently means that the original sentence is valid. This is possible by constructing possible worlds that will eventually falsify the sentence in case it is unsatisfiable. In case the sentence is satisfiable the algorithm may never terminate, as reasoning in FOL is semi-decidable. The method is still the basis for most modern automated proof methods.

For convenience, we consider sentences in so-called *Skolem normal form* (SNF). This means that all existential quantifiers are removed and all universal quantifiers are at the outer level of the sentence. Each sentence can be translated to a SNF, which is not necessarily equivalent, but is unsatisfiable if and only of the original sentence is unsatisfiable. Transforming the sentence such that all quantifiers are at the outer level, can be achieved by equivalence transformations. Removing the existential quantifiers is done by so-called *Skolemnisation*, which means that each existentially quantified variable is replaced by a *Skolem function* $f(X_1, \ldots, X_n)$, where f is some fresh function symbol and $X_1, \ldots, X_n$ are all universally quantified variables in the scope of the existential quantifier. Intuitively, the Skolem functions represent an arbitrary object satisfying the formula, of which there must be at least one in case the formula is satisfiable. We do not formally prove here that this transformation preserves unsatisfiability.

Example 2.4

Consider the following statement derived from Example 2.3, together with the assumption that there is a burglar in at least one house. Then there must also be an alarm in at least one house:

$$\big(\forall_H\ \mathit{burglary}(H) \rightarrow \mathit{alarm}(H)\big) \land \big(\exists_H\ \mathit{burglary}(H)\big) \rightarrow \big(\exists_H\ \mathit{alarm}(H)\big)$$

To show that this sentence is valid we can show that its negation is unsatisfiable:

$$\begin{aligned} &\neg\Big(\big(\forall_H\ \mathit{burglary}(H) \to \mathit{alarm}(H)\big) \land \big(\exists_H\ \mathit{burglary}(H)\big) \to \big(\exists_H\ \mathit{alarm}(H)\big)\Big) \\ \iff &\big(\forall_H\ \mathit{burglary}(H) \to \mathit{alarm}(H)\big) \land \big(\exists_H\ \mathit{burglary}(H)\big) \land \neg\big(\exists_H\ \mathit{alarm}(H)\big) \\ \iff &\big(\forall_H\ \mathit{burglary}(H) \to \mathit{alarm}(H)\big) \land \big(\exists_H\ \mathit{burglary}(H)\big) \land \big(\forall_H\ \neg\mathit{alarm}(H)\big) \end{aligned}$$

The above is obviously unsatisfiable. In case a burglar always causes an alarm ($\forall_H\ \mathit{burglary}(H) \to \mathit{alarm}(H)$) and there is at least one house with a burglar ($\exists_H\ \mathit{burglary}(H)$), then in this house there must be an alarm, which is contradicting with the statement that there is no alarm in any house ($\forall_H\ \neg\mathit{alarm}(H)$).

To formally show that, the first step it to obtain a SNF, by first bringing all quantifiers to the front:

$$\begin{aligned} &\big(\forall_H\ \mathit{burglary}(H) \to \mathit{alarm}(H)\big) \land \big(\exists_H\ \mathit{burglary}(H)\big) \land \big(\forall_H\ \neg\mathit{alarm}(H)\big) \\ \iff &\forall_H\ \exists_I\ \big(\mathit{burglary}(H) \to \mathit{alarm}(H)\big) \land \mathit{burglary}(I) \land \neg\mathit{alarm}(H) \end{aligned}$$

For the variable I we now introduce the Skolem function f and remove the existential quantifier:

$$\forall_H\ \big(\mathit{burglary}(H) \to \mathit{alarm}(H)\big) \land \mathit{burglary}\big(f(H)\big) \land \neg\mathit{alarm}(H)$$

The above sentence states that for each house H: (1) a burglar causes an alarm, (2) there is a burglar in some house, denoted by $f(H)$ and (3) there is no alarm in any house. This is unsatisfiable. There is no burglar in any house, as this would cause an alarm. So there cannot be a burglar in $f(H)$.

Note that in this case we could have also changed the order of the quantifiers, which means we would require only a *Skolem constant* instead of a 1-ary function. We did choose the order however for illustrational purposes.

The first step towards automated reasoning is to restrict the infinite number of possible domains of discourse, which is possible by using so-called *Herbrand universes*. In case a sentence is unsatisfiable under the Herbrand universe, this is true for all possible domains of discourse. The basic idea is to use terms built from applications of the functions appearing in the sentence themselves as elements of the domain of discourse. Each function application just represents itself. All information about the functions, that matters for reasoning, is encoded by the sentence. An interpretation assigning to all function applications the terms representing the function applications themselves, is called a *Herbrand interpretation*.

We now formally define the concept of Herbrand universes, as it will also play an important role in another kind of logic relevant for our work, called LP (Section 2.1.3). We start defining the set H_0, which includes all constants occurring in the sentence. In case no constants are included, we use a single arbitrary constant (e.g. $H_0 = \{a\}$). Further, for all $i > 0$, H_i is defined as the application of all occurring functions to the elements of H_{i-1}. The Herbrand universe is then defined as H_∞. So the Herbrand universe is infinite as soon as functions occur in the sentence.

Example 2.5

We continue with our example and repeat the SNF of the formula we have to prove unsatisfiability of:

$$\forall_H \, \big(burglary(H) \rightarrow alarm(H)\big) \wedge burglary\big(f(H)\big) \wedge \neg alarm(H)$$

As there is no constant we introduce a single arbitrary one, lets say a. The Herbrand universe is then constructed as:

$$\begin{aligned} H_0 &= \{a\} \\ H_1 &= \{a, f(a)\} \\ H_2 &= \{a, f(a), f(f(a))\} \\ &\ldots \\ H_\infty &= \{a, f(a), f(f(a)), f(f(f(a))), \ldots\} \end{aligned}$$

It remains the challenge that one has to consider all possible truth assignments to predicates for a potentially infinite domain. The first observation to solve that problem, is that in case a partial possible world falsifies the sentence, all possible worlds consistent with it, falsify the sentence as well. So a partial world can represent an infinite set of possible worlds. Each possible world is always in a set represented by a finite partial world falsifying the sentence, in case the sentence is unsatisfiable. This is the essence of *Herbrand's theorem* and actually means that unsatisfiability can be decided by generating finite partial worlds in an exhaustive and structured way until all partial possible worlds falsify the sentence. This may not terminate in case the sentence is satisfiable, which is equivalent to deciding whether a sentence follows from a theory, if the answer is negative. As discussed, this problem is only semi-decidable. For generating partial worlds a so-called *semantic tree* [131] can be used. Such tree splits at each level into the possible worlds, in which a specific predicate holds and does not hold respectively.

Example 2.6

To show that the sentence of the continued example is invalid, we construct partial interpretations as semantic tree, shown in Figure 2.2. At the first level, we choose the truth value for $burglary\big(f(a)\big)$. As the sentence requires that there is a burglar for all objects given by f for all possible arguments, the choice that there is no burglar for $f(a)$ falsifies the sentence. So we can close that branch and do not have to care about the interpretation of all other predicates.

In the other branch we know for sure that there is an alarm, which is implied by the fact that there is a burglar, because the sentence requires that $burglary(H) \rightarrow alarm(H)$ for all H. So choosing $alarm\big(f(a)\big)$ to be false, falsifies the entire sentence. Choosing $alarm\big(f(a)\big)$ to be true however also falsifies it, as the sentence requires that there is no alarm in any house. The tree exhaustively covers all possible interpretations and proves them to falsify the sentence, which proves that the sentence is unsatisfiable.

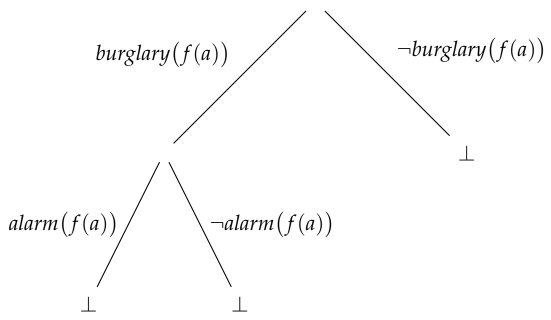


Figure 2.2: Semantic Tree Example

2.1.2.3 *Satisfiability Modulo Theories*

When building logical models, one often wants to use commonly applied theories, i.e. functions and predicates that already have a commonly used meaning. Examples are equality between the objects in the domain of discourse, but also inequalities and functions as addition and multiplication for integers and real numbers. Such problems are commonly referred to as *satisfiability modulo theories* (SMT) problems. The meaning of those functions could also be encoded in FOL, but using the intended objects, e.g. numbers, as domain of discourse for some variables is conceptually more clear and can be computationally more efficient.

Example 2.7

Suppose we also want to model the possibility that an earthquake causes the alarm and want to consider the earthquake's strength. We could classify the earthquake's strength in a discrete way, which means the domain of discourse could for instance be:

$$\mathbf{D} = \{no, low, medium, high, house_1, \dots, house_n\}$$

Then a possible theory would be:

$$\forall_H \ (burglary(H) \lor earthquake = high) \rightarrow alarm(H)$$

Here we make use of the function *earthquake* indicating the strength of the earthquake and also of the function *high*, which we assume to be the object from the domain of discourse with the same name in all possible worlds. Furthermore, we assume that equality is in all possible worlds defined as equality between objects of the domain of discourse.

We can also express the earthquake's strength by a continuous scale, then the domain of discourse could be:

$$\mathbf{D} = \mathbb{R} \cup \{house_1, \dots, house_n\}$$

A possibly theory could then look like:

$$\forall_H \ (burglary(H) \lor earthquake > 5.5) \rightarrow alarm(H)$$

Here we assume that *earthquake* only yields real numbers and not the house objects from the domain of discourse and further that $>$ has the common meaning for real numbers in all possible worlds. From *earthquake* > 8.0 one can still draw the conclusion that there is an alarm (actually in all houses), even though the number of remaining possible worlds is infinite, as the value of *earthquake* can be anything above 8.0.

Satisfiability is decidable for many relevant classes of theories, e.g. linear inequalities on integers (excluding multiplication) [116] and inequalities on real numbers including multiplication [36]. Also the complexity of deciding satisfiability is often manageable. For instance, the complexity of SMT solving is polynomial for equalities and inequalities on sums and differences of real numbers [45, 79]. Arithmetic with integers is however NP-complete [116], and real number arithmetic including multiplication is even more complex [36].

Such theories could as discussed in principle be encoded as FOL theories, but it is often much more efficient to use specialised algorithm for solving such problems. Such algorithms are included in so-called SMT solvers, e.g. *Yices* [46] and *Z3* [39].

2.1.2.4 *Discussion*

FOL is very useful in domains such as mathematics or where statements can be assumed to always hold. In many real world applications however virtually nothing can be said with absolute certainty. So either one underspecifies the problem, such that no useful conclusions can be drawn any more, or overspecifies it, such that observations can contradict the theory, making the new theory inconsistent and therefore useless. There are various **qualitative** solutions for such issues, for instance *default logic* [126], stating what *usually* holds given the opposite is not observed. We do not discuss those logics in detail, as we focus later on *probabilistic logic*, solving this problem in a **quantitative** way.

2.1.3 *Logic Programming*

LP is based on the idea of using predicate logic as a programming language [89]. *Prolog* is by far the most well-known example of such a language, although there are several other LP languages available. Since LP is based on FOL, the language is also very suitable as a basis for knowledge representation and reasoning.

2.1.3.1 *Syntax & Semantics*

A logic program **L** consists of a set of *rules*, also called *clauses*. Rules are (implicitly universally quantified) expressions of the form:

$$h \leftarrow l_1, \ldots, l_n$$

where h is called the *head* and the collection of *literals* $l_1, \ldots, l_n$ form the *body* of the rule and represent a conjunction. The head h is an *atom*, i.e. an expression of the form $p(t_1, \dots, t_m)$ with p a predicate and $t_1, \dots, t_n$ are terms, i.e. function applications. In LP the meaning of such applications is fixed to the Herbrand interpretations, which is motivated by the aim to provide efficient *inference*. If there are no variables in all terms $t_1, \dots, t_m$, the atom $p(t_1, \dots, t_m)$ is called *ground*. The literals of a body are atoms (e.g. a) or negated atoms (e.g. $not(a)$). *Facts*, also called *unit clauses*, are clauses without a body, assumed to be always true.

Although the syntax of LP is a subset of FOL, the semantics of LP and FOL differ. The difference emerges from the semantics of negations in LP, which, similar to the *closed world assumption*, states that a certain conclusion is false if it cannot be derived from the knowledge base. In contrast, in FOL a statement is only proved to be false if the negation of this statement is implied by the knowledge base, which means that some statements cannot be decided to be true or false. This difference allows LP to express non-ground inductive definitions, such as transitive closures, which is not possible in FOL [58].

The main consequence of this difference is that for a large class of programs there is only a single unique possible world, which is also often called model in this context. Programs $\mathbf{L}$ without negation are characterised by their smallest model $M(\mathbf{L})$, called the *least Herbrand model*, consisting of ground atoms entailed by the logic program $\mathbf{L}$, i.e. $a \in M(\mathbf{L})$ iff $\mathbf{L} \models a$. In case the $\leftarrow$s in the program are treated as implications, there are obviously multiple Herbrand models $M(\mathbf{L})_1, M(\mathbf{L})_2, \dots$ consistent with the program. The least Herbrand model is the intersection of all those models $M(\mathbf{L})_1 \cap M(\mathbf{L})_2 \cap \cdots$. This model matches the intuitively intended model, interpreting the rules as defining when the head is true and when it is false. It is true in case one of the bodies is true and false otherwise, in contrast to the FOL interpretation without a unique model where the truth of the head would be undetermined in case no body is true. The least Herbrand model can also be characterised by the program's completion in FOL, introduced by Clark [26]. Rules for the same head, such as

$$\begin{aligned} & h \leftarrow l_{11}, \dots, l_{1m} \\ & \dots \\ & h \leftarrow l_{n1}, \dots, l_{nk} \end{aligned}$$

are characterised by the completion

$$h \leftrightarrow (l_{11} \wedge \dots \wedge l_{1m}) \vee \dots \vee (l_{n1} \wedge \dots \wedge l_{nk})$$

which unambiguously defines the truth value of h in case the truth value of all body elements is known.

Also logic programs with negations often have a unique model. *Stratified* logic programs with negations, meaning that the program disallows certain combinations of recursion and negations, also have a unique least Herbrand model [3]. Non-stratified programs may still have a unique model in the form of non-partial *well-founded models* [150] or *stable models* [53]. Even though there are programs without such unique models, we assume that all programs used have such model.

The fact that a logic program always leaves only a single possible world, is illustrated in Figure 2.3. It does not make sense to show the possible worlds remaining with the model only without observations, as the truth value of each predicate has to be given either explicitly by adding it to the program or implicitly by not including it in the program.

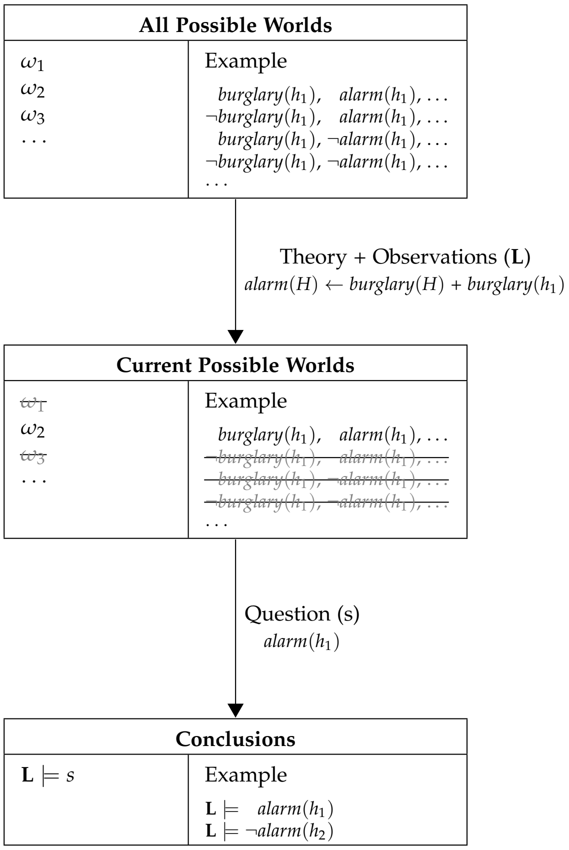

All Possible Worlds
ω_1
ω_2
ω_3
...
Example
$burglary(h_1)$, $alarm(h_1)$, ...
$\neg burglary(h_1)$, $alarm(h_1)$, ...
$burglary(h_1)$, $\neg alarm(h_1)$, ...
$\neg burglary(h_1)$, $\neg alarm(h_1)$, ...
...
Theory + Observations (**L**)
$alarm(H) \leftarrow burglary(H) + burglary(h_1)$
Current Possible Worlds
~~ω_1~~
ω_2
~~ω_3~~
...
Example
$burglary(h_1)$, $alarm(h_1)$, ...
~~$\neg burglary(h_1)$, $alarm(h_1)$, ...~~
~~$burglary(h_1)$, $\neg alarm(h_1)$, ...~~
~~$\neg burglary(h_1)$, $\neg alarm(h_1)$, ...~~
...
Question (s)
$alarm(h_1)$
Conclusions
$\mathbf{L} \models s$
Example
$\mathbf{L} \models alarm(h_1)$
$\mathbf{L} \models \neg alarm(h_2)$

Figure 2.3: Logic Programming Reasoning Illustration

Example 2.8

We illustrate Example 2.1 in the context of LP in Figure 2.3. One cannot leave out the observation, as not specifying that there is burglary means that we assume there is no burglary. Only the part of the possible worlds for the constant h_1 and the predicates *burglary* and *alarm* is given, because each possible world is infinite, as the domain of discourse ranges over all possible terms. Also the number of possible worlds itself is infinite. Given the program, actually all other grounded predicates are false. Therefore we can also conclude that there is no alarm for h_2.

If we would only observe that there is an alarm, we can conclude that there is no burglary, unlike as in FOL where in this situation the truth value of burglary is unknown. So here the usually made observation of an alarm may even lead to a possibly wrong conclusion, which illustrates the limited ability of LP to deal with uncertain situations.

2.1.3.2 *Constraint Logic Programming*

Constraint logic programming (CLP) [72] is the LP equivalent of SMT. It extends LP such that it can also deal with objects with associated theories as numbers, by also allowing special predicates (e.g. $<$, $\leq$, $\ldots$) on function applications of variables in rules, which actually *constrain* the possible values a variable can take. The general approach supports various constraint theories. For example, in CLP(FD) [27] constraints involving variables with a finite domain can be handled. Another example is CLP($\mathcal{R}$) [73], which introduces constraints on real numbers inside LP. Note that usually implementations of Prolog, without using CLP, also support numbers and special predicates as $<$. However, those predicates are evaluated by the time they occur during resolution, which only works in case the arguments are grounded numbers. This restricts the constraints on a numeric variable, as it must be fixed to a point and cannot only be constrained to an interval, and also makes the result of resolution dependent on the evaluation order, i.e. there is no declarative semantics when using those predicates.

Example 2.9

We go back to Example 2.7 and model the fact that an earthquake with strength above 5.5 also causes the alarm by CLP($\mathcal{R}$):

$$\begin{aligned}&\mathit{alarm}(H) \leftarrow \mathit{burglary}(H)\\&\mathit{alarm}(H) \leftarrow \mathit{earthquake}(E),\ \langle E > 5.5\rangle\end{aligned}$$

The brackets $\langle\rangle$ indicate that the predicates inside (in this case $<$) are not defined by the program, but have a special meaning given by a theory[1]. In case we observe that the earthquake has a strength above 8.0, we add the following to the program:

$$\mathit{earthquake}(E) \leftarrow \langle E > 8.0\rangle$$

From this we can conclude that there is an alarm in all houses.

2.1.3.3 *Discussion*

LP is very suited for cases in which one can **define** predicates in terms of other ones, therefore it is more similar to a programming language. It provides structure that makes definitions easier to capture than FOL definitions, as to understand the definition of a predicate one only has to inspect all rules with the

1 We use $\langle\rangle$ rather than $\{\}$ as usually used, to avoid confusion with set notation in mathematical definitions.

predicate as head. Furthermore, it allows to reason in a more efficient way and can represent knowledge not expressive in FOL (e.g. transitive closures) [58].

For models as the burglary model it is less suited, because one has to choose the direction in which the model is defined, i.e. whether burglary and earthquake are defined in terms of alarm or the other way around, which restricts the direction in which one can reason. There are **qualitative** solutions to this, such as *answer-set programming* [92, 112], which allows one to ask for the possible explanations given observations, e.g. an alarm. Probabilistic logic however again provides a solution, not bound to a direction of reason, in a **quantitative** way.

2.2 PROBABILITY THEORY

As discussed, traditional logic can deal with facts that are not certainly known, if at all, only in qualitative ways. Most real world problems however require dealing with uncertainty. Probability theory offers a widely used and well-founded basis for representing and reasoning with such uncertainty.

Dealing with probabilities as a scientific discipline, developed much slower than logic. This is in particular because is was long questioned that it makes sense at all to deal with random phenomena in a scientific way. Also there is a dispute about the meanings of probabilities. A first view, known as the *frequentist view*, is that probabilities are physical quantities, which can be measured by repeated observations. In the *Bayesian view*, probabilities represent the believe that something holds, given the information available. From a practical point of view, it matters less whether we do not know something because it is inherently random or because we do not have sufficient information. Probability theory is successfully applied for dealing with all kinds of uncertainty.

The mathematical foundations for probability theory were laid by Bernoulli [10] and de Moivre [38]. The modern theory of probabilities, also used in this thesis, was developed by Kolmogorov [81] and is in particular also suited for the case of continuous problems, meaning that there are uncountably many possible states of the world.

2.2.1 *Measures on Possible Worlds*

The theory, as logic, also makes use of the concept of possible worlds, which is here called the *sample space*. Probability theory provides a general way to deal with possible worlds of any structure. No structure of possible worlds is prescribed; the sample space can be an arbitrary set. Probability theory is basically about not only determining whether an *event* possibly occurs, but measuring how likely that is by a probability between 0 and 1, where 0 means that the event *(almost) certainly*[2] does not occur and 1 the opposite. There is also no struc-

2 Note that the difference between *certainly* and *almost certainly* has to be made for continuous distributions. For instance, the probability that the temperature is exactly 20 °C is 0.0, but theoretically possible. Therefore, we cannot say that this event does "certainly" not occur, but have to say that it does "almost certainly" not occur.

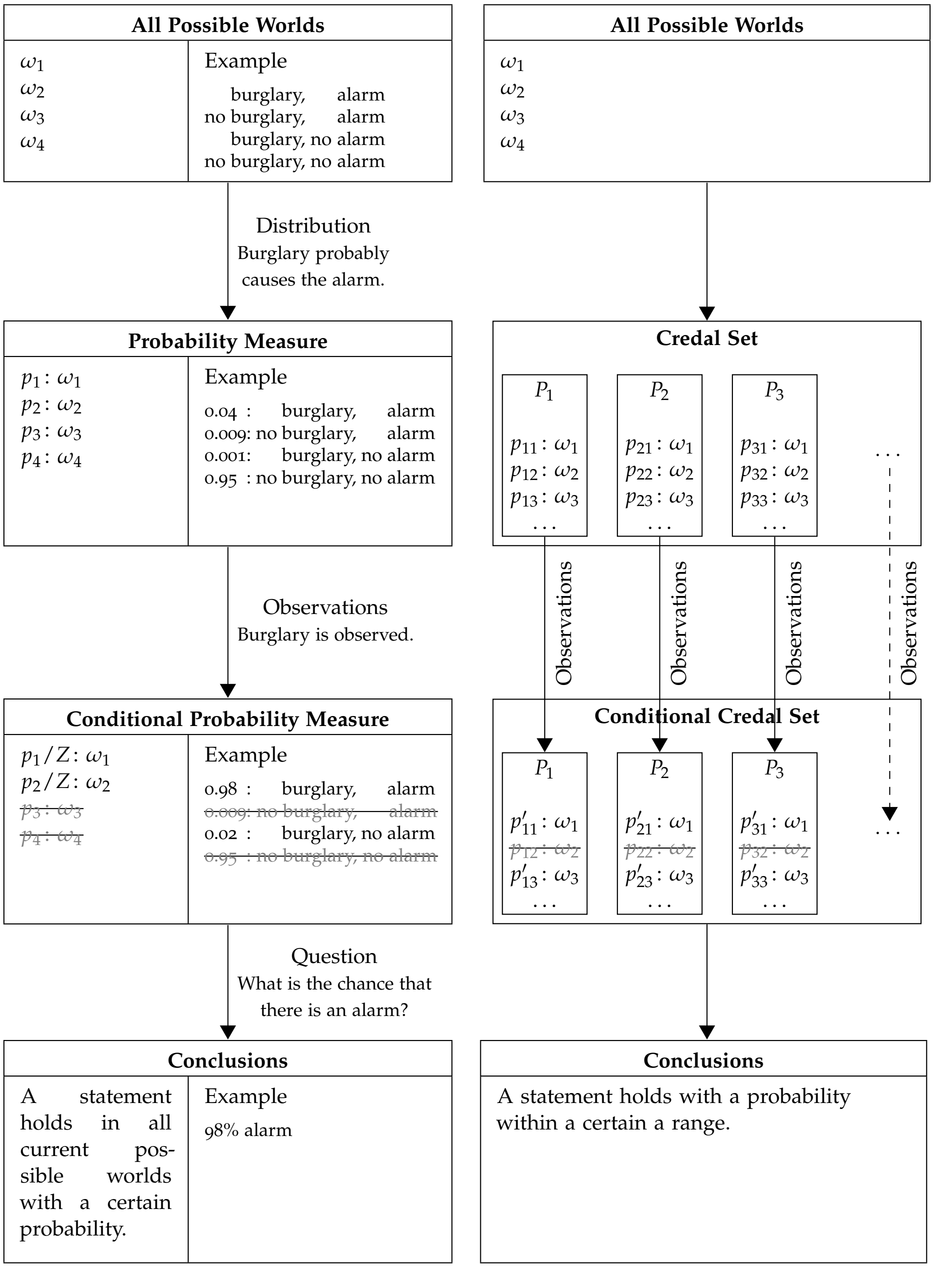
ture prescribed to represent such events, as is given in logic by the structure of sentences. Probabilistic events are just sets including possible worlds, meaning that the world is in one of the states in the set.

Probability theory provides well-defined ways to combine probabilities and makes use of partial knowledge about the state of the world by incorporating observations, in probability theory usually called *evidence*. Here we have to make a strict distinction between the model of the world and evidence, unlike in FOL. In FOL, statements which are not always true or false, e.g. whether there is a burglar, have to be left open and are then determined by observations. LP forces to determine whether something holds or not, so it does not make sense distinguishing between the model and observations. Probability theory does not force one to decide for the extremes, but instead one is forced to specify a probability, indicating how likely the event occurs. For instance, one has to say that there is a burglar with a chance of 0.001. If a burglar is observed this means that the probability of that event becomes 1.0, which is inconsistent with the model. So the combination of observations and model requires a special way of updating the probabilities, which is also called *conditioning*. One speaks about the probability of an event, given observed other ones. This conditioning actually works similar to adding sentences to a FOL theory: possible worlds inconsistent with the evidence are excluded. In probability theory additionally the probabilities have to be renormalised to make sure probability 1.0 is still assigned to the space of all possible worlds remaining.

This is illustrated in Figure 2.4a. The world is modelled by a *probability measure* over all possible worlds, we also call a *probability distribution*. The most straightforward way to define such a measure, is to assign a probability to each possible world, as is done in the illustration. However, this is not feasible for realistic models, as the number of probabilities grows exponentially and even impossible in case the space of possible worlds is uncountably large, e.g. in case real numbers are involved. This general case is discussed later. Possible worlds are excluded given observations and the probabilities of the remaining worlds are renormalised. Finally, one can reach conclusions about how likely a certain event is in the updated conditional measure.

Example 2.10

We consider again the burglary problem with only a single house, so there are only four possible worlds as in Example 2.1. The fact that burglary causes the alarm is expressed by the fact that the probability for the case that there is burglary, but no alarm, is low. This case is actually completely excluded by the FOL model (Figure 2.1b), but probability theory makes it possible to not completely exclude it, but to state that it is unlikely. One actually has to define probabilities for all cases, e.g. that there is a burglar and the likelihood of a false alarm. A possible probability assignment is given in Figure 2.4a. In this model, the probability that there is an alarm is only $0.04 + 0.009 = 0.049$. The low probability makes sense as hopefully there are not that many burglaries and false alarms.


All Possible Worlds
ω_1 ω_2 ω_3 ω_4
Example
burglary, alarm
no burglary, alarm
burglary, no alarm
no burglary, no alarm
Distribution
Burglary probably causes the alarm.
Probability Measure
$p_1 : \omega_1$ $p_2 : \omega_2$ $p_3 : \omega_3$ $p_4 : \omega_4$
0.04 : burglary, alarm
0.009: no burglary, alarm
0.001: burglary, no alarm
0.95 : no burglary, no alarm
Observations
Burglary is observed.
Conditional Probability Measure
$p_1/Z : \omega_1$ $p_2/Z : \omega_2$
0.98 : burglary, alarm
0.02 : burglary, no alarm
Question
What is the chance that there is an alarm?
Conclusions
A statement holds in all current possible worlds with a certain probability.
98% alarm
Credal Set
P_1 P_2 P_3
$p_{11} : \omega_1$ $p_{12} : \omega_2$ $p_{13} : \omega_3$
$p_{21} : \omega_1$ $p_{22} : \omega_2$ $p_{23} : \omega_3$
$p_{31} : \omega_1$ $p_{32} : \omega_2$ $p_{33} : \omega_3$
Observations
Conditional Credal Set
$p'_{11} : \omega_1$ $p'_{13} : \omega_3$
$p'_{21} : \omega_1$ $p'_{23} : \omega_3$
$p'_{31} : \omega_1$ $p'_{33} : \omega_3$
A statement holds with a probability within a certain a range.

(a) Precise Probabilistic Reasoning (b) Imprecise Probabilistic Reasoning

Figure 2.4: Probabilistic Reasoning Illustrations

In case we then observe a burglar, as in the FOL example (Figure 2.1b) we exclude the worlds without burglary. However, we also renormalise the probabilities, so the probability of alarm raises to 0.98, which makes sense even if before the probability for an alarm was that low, because in the unlikely possible worlds in which there is a burglar, there is also likely an alarm. This is illustrated in Figure 2.4a. One can also without changing the model, compute the probability of burglary given an alarm in the same way, i.e. reverse the direction of reasoning, which is a big advantage of probability theory.

As soon as the sample space becomes uncountably large, a measure cannot directly be defined on it. It is not possible to define a measure on arbitrary spaces. Measure theory requires that such spaces are *measurable*. This means that it is in general impossible to assign meaningful measures to all possible events. Therefore, the events are restricted by means of a so-called *event space*. We introduce now this very general, measure-theoretic approach to probability theory, specifically Kolmogorov's probability theory [81]. Often a simplified view is used for AI models, but the general theory is necessary for the work presented in this thesis, as it is concerned with constructing complex probability distributions.

Probability spaces form the basis of modelling uncertain processes in the theory presented. A probability space $(\Omega, \mathcal{A}, P)$ formally consists of:

1. A *sample space* Ω: An arbitrary set of all possible states the world can be in.
2. An *event space* $\mathcal{A}$: A subset of the sample space's powerset ($\mathcal{A} \subseteq \wp(\Omega)$) which is a *σ-algebra*, meaning that it (*i*) contains the empty set ($\emptyset \in \mathcal{A}$), is closed under (*ii*) complement ($e \in \mathcal{A} \implies (\Omega \setminus e) \in \mathcal{A}$) and (*iii*) countable union ($e_1, e_2 \in \mathcal{A} \implies (e_1 \cup e_2) \in \mathcal{A}$). A consequence of properties (*i*) and (*ii*) is that the entire sample space Ω is always part of the event space as well. Elements of $\mathcal{A}$ are called events and probabilities are only assigned to these events.
3. A probability measure P: A function assigning a number from the closed interval $[0, 1]$ to any event. P must be *countably additive*, which means that the probability of the union of countably many pairwise disjoint events must equal the sum of the probabilities of those events ($P(e_1 \cup e_2) = P(e_1) + P(e_2)$, if $e_1 \cap e_2 = \emptyset$). Additionally, P must assign 1 to the entire sample space ($P(\Omega) = 1$).

Example 2.11

The sample space of the burglary example can be defined by a tuple of two boolean values:

$$\Omega = \{\ (\mathit{true}, \mathit{true}), (\mathit{false}, \mathit{true}), (\mathit{true}, \mathit{false}), (\mathit{true}, \mathit{true})\ \}$$

Here the first value represents whether there is a burglar and the second whether there is an alarm. The event space can in this example just be the powerset of Ω, which is always a σ-algebra for countable sample spaces. The probability measure can be defined as:

$$P\big((\mathit{true}, \mathit{true})\big) = 0.04 \qquad P\big((\mathit{false}, \mathit{true})\big) = 0.009$$
$$P\big((\mathit{true}, \mathit{false})\big) = 0.001 \qquad P\big((\mathit{false}, \mathit{false})\big) = 0.95$$

This uniquely defines a probability measure P, i.e. establishes probabilities for all other events in the event space, given the properties of a probability measure. The probability for an alarm is computed as:

$$P\Big(\{(\mathit{true}, \mathit{true}), (\mathit{false}, \mathit{true})\}\Big) = 0.04 + 0.009 = 0.049$$

In case we want to extend the model with the strength of an earthquake, measured as real number, we extend the previously defined space: $\Omega \times \mathbb{R}$. The powerset of this space is not measurable; one has to choose a proper event space, which is for instance possible by considering all real intervals with rational bounds as events. Also the probability measure cannot be defined by assigning probabilities to every possible world any more. How such measures are defined is discussed later.

We next define some key concepts, which are important to achieve a compact representation and efficient inference. The concept of *independence* is especially important for efficient inference. Two events e and f are independent in case the occurrence of the one does not affect the other. In that case:

$$P(e \cap f) = P(e)P(f) \tag{2.1}$$

That e and f are independent is denoted as:

$$e \perp\!\!\!\perp f$$

Another property essential for efficient inference, directly given by the definition of probability space before, is that for two mutually exclusive events ($e \cap f = \emptyset$) holds:

$$P(e \cup f) = P(e) + P(f) \tag{2.2}$$

Conditional probabilities are probabilities conditioned on the knowledge that an event is certainly true. The probability of event e given f for which $P(f) > 0$ is defined as:

$$P(e \mid f) \overset{\text{def}}{=} \frac{P(e \cap f)}{P(f)} \tag{2.3}$$

The given event f is also referred to as evidence.

Example 2.12

We can compute the conditional probability of an alarm given that there is a burglar as:

$$P\Big(\{(\mathit{true}, \mathit{true}), (\mathit{false}, \mathit{true})\} \mid \{(\mathit{true}, \mathit{true}), (\mathit{true}, \mathit{false})\}\Big)$$
$$= \frac{P\Big(\{(\mathit{true}, \mathit{true})\}\Big)}{P\Big(\{(\mathit{true}, \mathit{true}), (\mathit{true}, \mathit{false})\}\Big)} = \frac{0.004}{0.0041} \approx 0.98$$

Analogously, we can compute the probability of a burglar given an observed alarm:

$$P\Big(\big\{(\mathit{true},\mathit{true}),(\mathit{true},\mathit{false})\big\} \mid \big\{(\mathit{true},\mathit{true}),(\mathit{false},\mathit{true})\big\}\Big)$$
$$=\frac{P\Big(\big\{(\mathit{true},\mathit{true})\big\}\Big)}{P\Big(\big\{(\mathit{true},\mathit{true}),(\mathit{false},\mathit{true})\big\}\Big)} = \frac{0.004}{0.0049} \approx 0.82$$

2.2.2 *Random Variables*

Often *random variables* are used to represent probability distributions and we denote them with bold upper-case letters, e.g. $\mathbf{X}$, $\mathbf{Y}$. We assume that each random variable $\mathbf{V}_i$ has a range $\mathbf{Range}_i$ of values. Sometimes sets of random variables are indicated by an index set: $\mathbf{V}_I = \{\mathbf{V}_i \mid i \in I\}$, where I is an index set. Random variables are actually functions mapping the sample space to the variable's range: $\mathbf{V}_i \colon \Omega \to \mathbf{Range}_i$. This has to be done in such a way that the probability measure P of the original probability space assigns a probability to each event concerning the random variable $\mathbf{V}_i$ as well. We usually use a simple mapping from sample spaces to random variables. We represent the random variables' values as tuples of which each element represents a single random variable's state. The random variables are therefore just functions mapping a tuple to one specific element.

A distribution assigning probabilities to values assignments of a number of random variables is called a *joint distribution*. In the case of a finite number of random variables with finite ranges, one can uniquely define a joint probability distribution P by assigning a probability to each joint assignment of values $v_i \in \mathbf{Range}_i, i = 1, \ldots, n$, to random variables $P(\mathbf{V}_1 = v_1, \ldots, \mathbf{V}_n = v_n)$. Such a function, assigning probabilities to values of random variables, is also called *probability mass function* (PMF). From probability distributions one can compute the probability of partial assignments using *marginalisation*:

$$P(\mathbf{V}_K = v_K) = \sum_{v_J, J = I \setminus K} P(\mathbf{V}_K = v_k, \mathbf{V}_J = v_J), \tag{2.4}$$

where $I = \{i \mid 1 \leq i \leq n\}$.

Example 2.13 ___

Meaningful random variables for the burglary example, where the first element of the sample space represents whether there is a burglar and the second whether there is an alarm, can be defined as: $\mathbf{Burglary}\big((\omega_1, \omega_2)\big) = \omega_1$ and $\mathbf{Alarm}\big((\omega_1, \omega_2)\big) = \omega_2$. We can also express the distribution of Example 2.10 as:

$$P(\mathbf{Burglary} = \mathit{true}, \mathbf{Alarm} = \mathit{true}) = 0.04$$
$$P(\mathbf{Burglary} = \mathit{false}, \mathbf{Alarm} = \mathit{true}) = 0.009$$
$$P(\mathbf{Burglary} = \mathit{true}, \mathbf{Alarm} = \mathit{false}) = 0.001$$
$$P(\mathbf{Burglary} = \mathit{false}, \mathbf{Alarm} = \mathit{false}) = 0.95$$

We can use marginalisation to compute for instance:

$$
\begin{aligned}
&P(\textbf{Alarm} = \textit{true}) \\
=&P(\textbf{Burglary} = \textit{true}, \textbf{Alarm} = \textit{true}) + P(\textbf{Burglary} = \textit{false}, \textbf{Alarm} = \textit{true}) \\
=&0.04 + 0.009 = 0.049
\end{aligned}
$$

This works well for the discrete case, but the continuous case, i.e. if random variables have uncountable ranges, requires more sophisticated techniques. In case the range of random variables are the real numbers, probability measures for intervals are often defined in terms of *probability density functions* (PDFs). If f is a PDF of a random variable $\mathbf{V}_i$ then the probability that $\mathbf{V}_i$ takes a value within the interval $[a, b]$ is defined as:

$$P\big(\mathbf{V}_i \in [a, b]\big) \overset{\text{def}}{=} \int_a^b f(x)\,\mathrm{d}x \tag{2.5}$$

The main problem with that definition is that integrals can only be computed exactly or even be approximated with certain and acceptable error in very limited cases.

Generally, random variables offer an intuitive way to look at probability distributions, although they are not always suited to construct complex probability distributions.

2.2.3 *Imprecise Probability Theory*

Imprecise probability theory is a generalisation of probability theory. It avoids using crisp probabilities and therefore allows one to express ignorance concerning probability distributions. The theory is used if it is not possible to obtain precise probabilities, either due to insufficient availability of data to estimate the probabilities, or because probabilities are estimated by a number of experts, thus providing a range of probabilities. Imprecise probabilities are discussed here, because many methods to combine probability theory and logics, also the work in this thesis, do not define a single probability measure, but actually only put constraints on such measure, which leads to imprecise probabilities.

There are different approaches to imprecise probabilities with varying expressiveness [154]. In this thesis we consider convex sets of probability distributions, referred to as *credal sets* [87]. This setting makes it possible to express probabilities of events by a lower and upper bound. Formally, we denote a credal set as $\mathbf{P}$, assume it has an associated sample and event space, and consists of a collection of probability measures consistent with Kolmogorov's axioms. Such credal set is illustrated in Figure 2.4b. For each event e there exists a distribution in $\mathbf{P}$ for which the probability attains a minimum and maximum. We denote these probabilities by $\underline{P}(e)$ and $\overline{P}(e)$, respectively. Note that all probabilities in this interval are possible and none is preferred, thus with imprecise probabilities one can express ignorance about what the real probability is. In contrast, in the alternative method of using second-order probabilities, a probability distribution is defined for the original probability.

There are multiple definitions of conditional probabilities for imprecise probability distributions. Weichselberger argues that different notions of conditional probabilities should be used depending on the purpose they are used for [157]. We here restrict to what Weichselberger calls the *intuitive concept*. A credal set can then be conditioned similarly to single probability measures. This is done by conditioning all probability measures in the credal set, as illustrated in Figure 2.4b.

Example 2.14

Consider the two possible probability distributions for the burglary example in Table 2.2.

P_1	alarm	no alarm
burglary	0.004	0.001
no burglary	0.009	0.95

P_2	alarm	no alarm
burglary	0.03	0.02
no burglary	0.01	0.94

Table 2.2: Example Possible Probability Distributions in Credal Set

Actually, convex credal sets usually consist of infinitely many probability distributions, except for the corner case of a single point distribution. If we observe that there is a burglar, all distributions are updated accordingly, as shown in Table 2.3. We again only show two possible distributions.

P_1	alarm	no alarm
burglary	0.8	0.2
~~no burglary~~	~~0.009~~	~~0.95~~

P_2	alarm	no alarm
burglary	0.6	0.4
~~no burglary~~	~~0.01~~	~~0.94~~

Table 2.3: Conditioned Example Possible Probability Distributions in Credal Set

If we assume the two probability distributions remaining, after incorporating the evidence, represent the extreme points, we can conclude that the probability for an alarm is between 0.6 and 0.8. As we assume credal sets are convex, in this example the credal set also contains distributions assigning all infinitely many possible probabilities in between.

A price to pay for imprecise probabilities is usually an increased inference complexity. Very often, imprecise probabilistic inference is in general much more complex than precise probabilistic inference. For instance, obtaining a bound on a marginal probability given a *locally defined credal network* (LDCN) [31], an imprecise version of *Bayesian networks* (BNs), is $\mathsf{NP}^{\mathsf{PP}}$-complete, while it is PP-complete for BNs [37].

2.2.4 *Discussion*

Probability theory provides the necessary basis for dealing with uncertainty in a well-founded way. Imprecise probability theory relieves from the necessity to always exactly specify a probability distribution. It adds the possibility of expressing ignorance, which is also possible in classical FOL. The main drawback of probability theory is that it lacks structure as provided in logic by the logical languages. Random variables are a concept to provide a convenient view on probability distributions, but do not allow to specify probability distributions in a compact way. This requires additional means to express structure such as independencies and first-order abstraction.

3 PROBABILISTIC LOGICS

To practically build probabilistic models, one has to basically perform two steps: First, one has to define a measurable *event space*. Second, one has to define a measure (or a *credal set* of those) on this event space. As it is infeasible to build realistic models by straightforwardly defining a measure, representation languages are employed. They allow for a compact representation of distributions, by making assumptions about the probability distribution's structure. A crucial issue for making *probability theory* practical is therefore how the structure of probability distributions is expressed. This always requires a balance between convenience of model construction, expressiveness and efficiency of *inference*. The most popular formalisms in *artificial intelligence* (AI) use graphs to express distributions' structures. One of the most influential of such formalisms are *Bayesian networks* (BNs) [119], expressing dependencies between *random variables* using a directed, acyclic graph. However, such formalisms have the major shortcoming that they are *propositional*. This hinders a compact representation of many problems, similar to the shortcoming of propositional *logic* compared to *first-order logic* (FOL). So probabilistic formalisms should also be first-order.

From a computing science point of view, this can be achieved by providing probabilistic versions of programming languages, as all deterministic programming languages employed to solve realistic problems are at least first-order. There are various paradigms for probabilistic programming, which are related to paradigms for non-probabilistic programming. They range from graphical approaches (e.g. *multi-entity Bayesian networks* [84]), imperative and object-oriented approaches (e.g. *FACTORIE* [94], *Figaro* [120]), purely functional approaches (e.g. *Church* [57]), *logic programming* (LP) approaches (e.g. *independent choice logic* (ICL) [122], *ProbLog* [125]), to other logic approaches (e.g. *BLOG* [102]). We focus here on logic based approaches. The reason for this is first, that logic is very suited to study programming methods on a conceptual level. Furthermore, logic is widely applied for and very suited for knowledge representation, in contrast to imperative and functional approaches. The goal of this section is not so much to discuss details about choices made in the design of concrete languages, but more to contrast approaches at a conceptual level to make clear where the work in this thesis is positioned.

3.1 INTERNAL & EXTERNAL PROBABILISTIC LOGICS

Early work on *probabilistic logics* by Bacchus [6] and Halpern [63] was aiming at a logic to reason about probabilities in the most general way possible. This means that it is not only possible to express knowledge about concrete domains, but also to prove general theorems about probability theory, e.g. :

$$\forall_e \forall_f \ P(e \cap f) = 0.0 \rightarrow P(e \cup f) = P(e) + P(f)$$

Such languages are referred to as *internal probabilistic logics* by Williamson [158]. The *probability measure P* is embedded in such logics. It is a function about which properties can be stated and proven in the logic itself. This generality has the major drawback that validity is undecidable for such languages, which is not surprising as validity in FOL is undecidable as well. However, in contrast to FOL, such probabilistic logics are not even completely axiomatisable, so validity is not even semi-decidable as validity in FOL [1]. Issues with decidability can be avoided by assuming a finite *domain of discourse*, which makes such probabilistic logics decidable [1], still reasoning can be very complex. Most languages aiming to provide a modelling language for AI problems, externalise the probabilistic aspect, which means that the probability measure is not part of the logic itself. Instead, the properties of the probability measure that can be expressed, are restricted to making statements about probabilities of *events* in some external way. We focus on such *external probabilistic logics* [158] in the following.

For such external probabilistic logicss, the most important function of logic is to provide a mean to represent the *sample space* and events in a structured way. The key insight here is that logical structures naturally correspond to elements of probabilistic sample spaces. Both describe a possible state of the world; as said, we use the term *possible world* for the unification of both concepts. The only difference is that logical possible worlds provide more structure, as they assign values to named *predicate* and function symbols. As the sample space can be arbitrary, the set of all possible structures, or a subset hereof, can be used as sample space in the definition of a *probability space*.

Analogously, logic provides a structured way to define event spaces by means of sentences. This idea goes back to Boole, who already proposed to speak of the probability of propositions asserting the occurrence of an event, instead of the event itself [15, 62]. The connection to measure-theoretic probability theory was later drawn by Gaifman [51]. Sentences can be related to a set of possible worlds, that is the set of worlds which are models of the sentence. Formally, an event e_s corresponding to a sentence s is defined as:

$$e_s \overset{\text{def}}{=} \{\omega \in \Omega \mid \omega \models s\} \tag{3.1}$$

It makes sense to write $\omega \models s$ as Ω is a set of possible worlds. From now on we use a sentence to denote the event it represents if the meaning is clear from the context and also speak of the probability of sentences. So an event space can be defined as some subset of all possible sentences. The only technical pitfall here is that this set has to be a *σ-algebra* (see Section 2.2.1), which is however hardly an issue in practice, as the logic's signature is usually countable.

3.2 CONSTRAINTS ON PROBABILITY DISTRIBUTIONS

The most straightforward choice for an external probabilistic logics is to allow probabilistic constraints on arbitrary sentences. For propositional logic an early proposal for such logic is by Gaifman [51] and was picked up in the context of AI by e.g. Nilsson [113].

3.2.1 *Constraining Probabilities of Sentences*

Basically, as a logical theory is defined by a set of sentences, a theory in such probabilistic logic is defined by a set of constraints of sentences' probabilities:

$$\begin{aligned} &P(s_1) = p_1 \\ &\cdots \\ &P(s_n) = p_n \end{aligned} \tag{3.2}$$

This can be generalised to probability intervals (e.g. $P(s_i) \in [\underline{p_i}, \overline{p_i}]$). It can then be asked whether a probabilistic constraint on arbitrary sentences holds given such theory **T**, analogous to ordinary propositional logic:

$$\mathbf{T} \models P(s \mid \mathbf{O}) \in [\underline{p}, \overline{p}] \tag{3.3}$$

The approach is illustrated in Figure 3.1a. We start from all possible worlds, which are in this case possible structures of a logical theory. From that we can consider all possible probability measures on those possible worlds. The theory, consisting of statements as in Equation 3.2, then restricts the possible measures, in the same way as discussed for general *imprecise* probability theory (Section 2.2.3). The set of possible worlds left over is actually a credal set about which conclusions can be drawn.

Example 3.1 ____

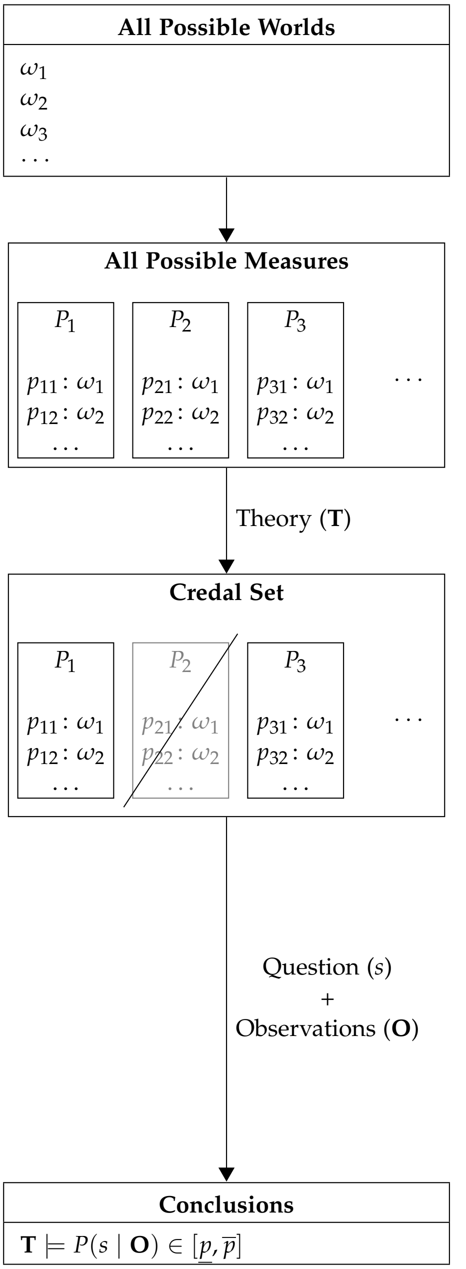
Coming back to the burglary example we could define a theory:

$$\begin{aligned} P(burglar) &= 0.02 \\ P(burglar \rightarrow alarm) &= 0.99 \end{aligned}$$

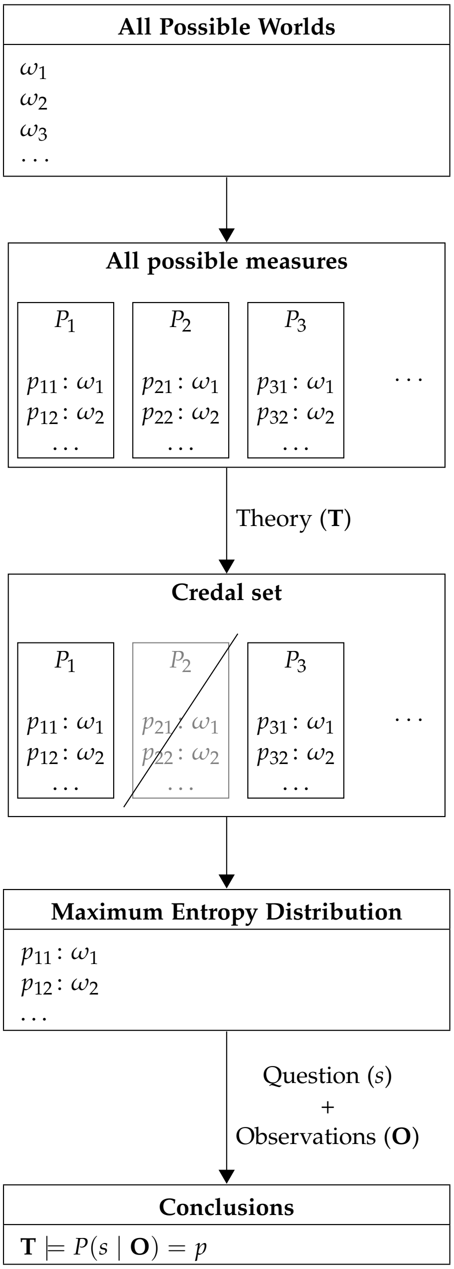
From this one can conclude:

$$0.01 \leq P(alarm) \leq 0.99$$

This conclusion may seem counter-intuitive at first, but is clarified by Table 3.1, which illustrates the distributions with the extreme values for the probability of *alarm*. The only world not consistent with *burglar* → *alarm* is the world in which


All Possible Worlds
ω_1
ω_2
ω_3
...
All Possible Measures
P_1 $p_{11}: \omega_1$ $p_{12}: \omega_2$ ...
P_2 $p_{21}: \omega_1$ $p_{22}: \omega_2$ ...
P_3 $p_{31}: \omega_1$ $p_{32}: \omega_2$ ...
...
Theory (**T**)
Credal Set
P_1 $p_{11}: \omega_1$ $p_{12}: \omega_2$ ...
P_2 $p_{21}: \omega_1$ $p_{22}: \omega_2$ ...
P_3 $p_{31}: \omega_1$ $p_{32}: \omega_2$ ...
...
Question (s) + Observations (**O**)
Conclusions
$\mathbf{T} \models P(s \mid \mathbf{O}) \in [\underline{p}, \overline{p}]$

(a) Constraints on Probability Distributions


All Possible Worlds
ω_1
ω_2
ω_3
...
All possible measures
P_1 $p_{11}: \omega_1$ $p_{12}: \omega_2$ ...
P_2 $p_{21}: \omega_1$ $p_{22}: \omega_2$ ...
P_3 $p_{31}: \omega_1$ $p_{32}: \omega_2$ ...
...
Theory (**T**)
Credal set
P_1 $p_{11}: \omega_1$ $p_{12}: \omega_2$ ...
P_2 $p_{21}: \omega_1$ $p_{22}: \omega_2$ ...
P_3 $p_{31}: \omega_1$ $p_{32}: \omega_2$ ...
...
Maximum Entropy Distribution
$p_{11}: \omega_1$
$p_{12}: \omega_2$
...
Question (s) + Observations (**O**)
Conclusions
$\mathbf{T} \models P(s \mid \mathbf{O}) = p$

(b) Maximum Entropy Distribution

Figure 3.1: Probabilistic Logic Languages

holds $burglar \wedge \neg alarm$, so its probability is fixed to $1.0 - 0.99 = 0.01$. Because of the constraint $P(burglar) = 0.02$, we also have to fix the probability of the world $burglar \wedge alarm$ to 0.01. The theory does not constrain how the remaining probability mass is distributed between the two remaining possible worlds and the two choices corresponding to the extreme probabilities for *alarm* are given in Table 3.1.

	alarm	*¬alarm*
burglar	0.01	0.01
¬burglar	0.0	0.98

	alarm	*¬alarm*
burglar	0.01	0.01
¬burglar	0.98	0.0

Table 3.1: Probability Distributions With Extreme Points for $P(alarm)$

3.2.2 *First-Order Definitions*

Clearly, employing logic employing at least first-order variables, is desirable, as this makes such logic more expressive than other propositional ways to construct probability distributions, such as graphical models. The approach discussed for propositional logic can in principle be transferred to FOL [51]. However, for expressing general, statistical knowledge, which one naturally expects to be expressable in a probabilistic FOL, a different kind of semantics is required, as pointed out by Bacchus [6] and Halpern [63]. They developed probabilistic logics with a special quantified probability measure, representing the probability given a randomly selected object for the quantified variable.

The semantics of this general approach is quite different from the semantics based on measures on possible worlds, as discussed so far. Instead, it is based on measures on the domain. Despite of this difference in semantics, it has been shown that such logics can be combined with logics based on probabilities on possible worlds and that both logics are in fact equally expressive [1]. Most practical languages discussed later on allow to specify the same *independent* probability for all objects. So given a predicate *pred* one can specify that the probability that $pred(X)$ holds is some probability p for all objects filled in for X. This is a pragmatic choice and not the same as specifying that p is the probability that *pred* holds for a randomly chosen object. The first problem with this method is that adding a different probability that *pred* holds for a specific object makes the theory inconsistent. Still, one can incorporate observations about specific objects by conditioning on them. A second, more theoretical issue is that the method does not support *learning in the limit*. Learning in the limit means that if a property is observed for a number of objects, this should induce the universal statement that this property holds for all objects, in case the number of objects with the observed property goes to infinity. In the settings described above this fails as the probability of infinitely many observations with independent probabilities, converges to 0.0. This issue and a possible solution is discussed by Hutter et al. [70] in detail. Here we restrict to the discussion of more pragmatic solutions, which the goal of practical applicability.

Example 3.2

Consider the statement: "99% of the time a burglar causes an alarm." A first failed attempt to formalise this may be:

$$P\big(\forall_H \; burglary(H) \to alarm(H)\big) = 0.99$$

This is however not a correct formalisation, as it actually means that with probability 0.99 a burglar causes an alarm for all houses. Possible worlds where only a single alarm system fails, are inconsistent with that statement.

One could move the quantifier outside to match the intuition that this probability holds for all houses separately:

$$\forall_H \; P\big(burglary(H) \to alarm(H)\big) = 0.99$$

which has to be understood as abbreviation for:

$$\begin{aligned}&P\big(burglary(h_1) \to alarm(h_1)\big) = 0.99\\&P\big(burglary(h_2) \to alarm(h_2)\big) = 0.99\\&\ldots\end{aligned}$$

This is problematic in case we add knowledge about a particular house:

$$P\big(\neg burglary(h_1)\big) = 0.999$$

The sentence $\neg burglary(h_1)$ represents a subset of the possible worlds represented by the implication $burglary(h_1) \to alarm(h_1)$. Therefore, it cannot have a larger probability than this implication. This problem is often circumvented practically by considering observations about specific objects as *evidence*, updating the probability measure, thus causing no inconsistency.

The aim of Bacchus was however to design a general language for reasoning about probabilities without those restrictions [6]. In this language, the knowledge can then be formalised as:

$$[burglary(H) \to alarm(H)]_H = 0.99$$

This means that the probability that the implication holds for a randomly selected house H is 0.99, which corresponds to the statement we aim to formalise.

Just as, given a FOL theory there are sentences of which we cannot decide whether they hold or not, given probabilistic constraints on sentences there are probabilistic constraints on sentences of which we cannot decide whether they are true or not. This means probabilistic constraints on sentences define a credal set rather than a single probability measure, although the connection with imprecise probability theory is usually not drawn in such work on probabilistic logics. It is however often considered to be desirable to define a single probability measure for various reasons. First, in case one bases decisions on such distribution, there is a single best decision in the precise, but not always the imprecise case. Second, inference and learning are much easier for precise distributions, as discussed in Section 2.2.3. Finally, the credal set, a probabilistic FOL theory defines, may also be empty, similar to the fact that an ordinary FOL theory may be inconsistent. This is often problematic when incorporating new knowledge. Analogously, for non-probabilistic logic languages a large portion of work is devoted to avoiding inconsistency in theories after adding new knowledge. So also most recent work on probabilistic logics aims at providing a formalism in which each theory defines a single probability distribution.

3.2.3 *Continuous Distributions*

The idea of probability constraints can also straightforwardly be applied to continuous distributions, which means that the number of possible worlds becomes uncountably infinite. Surprisingly, we found no previous work considering such extension before and such extension is a first step towards the concept proposed in this thesis. The extension is based on the observation that we can reason about probabilities on infinitely many possible worlds, in a similar way as reasoning in FOL, where we can also reason about an infinite number of possible worlds. It is however not possible to define a unique probability measure and remain decidable, as this requires assigning infinitely many probability values.

Example 3.3

Consider the statement:

$$P(\mathit{earthquake} > 8.0) = 0.9 \land P(\mathit{earthquake} > 5.5 \rightarrow \mathit{alarm}) = 0.95$$

As we do not define a single probability measure, we do not have to define the distribution in terms of *probability density functions* (PDFs). Because of this, we can conclude $0.85 \leq P(\mathit{alarm}) \leq 0.95$ without solving an integral, even though *earthquake* has uncountably many possible values. Similar as for Example 3.1, the possible distributions with extreme probabilities for *alarm* are illustrated in Table 3.2. There are many possible distributions with the same maximal probability for *alarm* and we give two representatives. Blank fields mean that the remaining probability can arbitrarily be distributed between all of them.

3.3 MAXIMUM ENTROPY MODELS

A possible way to restrict to a single distribution is to pick the one with the maximal *entropy*. This of course requires consistent definitions as starting point, such that there are distributions to choose from. The distribution with the maximal entropy makes the least assumptions, additional to the ones given by the probabilistic constraints. This is a general technique for selecting a unique probability measure in case the measure is not completely determined. Some justifications for the choice of the maximum entropy distribution are given by Paris [117]. The

	earthquake $\in (-\infty, 5.5]$	*earthquake* $\in (5.5, 8.0]$	*earthquake* $\in (8.0, \infty)$
alarm	0.0	0.0	0.85
$\neg$*alarm*	0.1	0.0	0.05

	earthquake $\in (-\infty, 5.5]$	*earthquake* $\in (5.5, 8.0]$	*earthquake* $\in (8.0, \infty)$
alarm			0.85
$\neg$*alarm*	0.0	0.0	0.05

	earthquake $\in (-\infty, 5.5]$	*earthquake* $\in (5.5, 8.0]$	*earthquake* $\in (8.0, \infty)$
alarm			0.9
$\neg$*alarm*	0.0	0.05	0.0

Table 3.2: Probability Distributions With Extreme Points for $P(alarm)$

idea is illustrated in Figure 3.1b. As before a theory restricts the possible distributions to form a credal set of which then the distribution with the maximal entropy is selected.

In this section we assume a finite number of possible worlds. The general, infinite case will shortly be discussed later in Section 3.3.3. A finite number of possible worlds can be imposed by assuming a finite number of predicates, functions with known definition and finitely many objects in the domain of discourse. The distribution can then be defined by a *probability mass function* (PMF) P on the possible worlds and its entropy is then defined by:

$$H(P) \stackrel{\text{def}}{=} -\sum_{\omega\in\Omega} P(\omega)\log\big(P(\omega)\big) \tag{3.4}$$

The basic idea is to choose the distribution with maximum entropy, which is at the same time consistent with the constraints given (Equation 3.2). Nilsson [113] already proposed to choose a unique probability measure in this way. This is however not straightforward in practice, as it is hard to determine such distribution.

3.3.1 *Maximum Entropy Probabilistic Logic*

Paskin [118] first proposed a probabilistic logic, which makes finding the maximum entropy distribution computationally feasible. The idea is to use *log-linear models*, for which it is feasible to find a maximum entropy distribution. The sentences probabilistic constraints are put on $s_1, \ldots, s_n$ are used as so-called *features* in such a model. The distribution is then defined as:

$$P(\omega) \overset{\text{def}}{=} \frac{\exp\left(\sum\limits_{i=1}^{n} w_i \mathbb{1}_{s_i}(\omega)\right)}{\sum\limits_{\omega \in \Omega} \exp\left(\sum\limits_{i=1}^{n} w_i \mathbb{1}_{s_i}(\omega)\right)} \tag{3.5}$$

Here $\mathbb{1}_s$ is an indicator function, giving 1 in case sentence s holds in the world given as argument. The model is defined by weights $w_1, \ldots, w_n$ assigned to sentences. There is a weighted feature for all and only those sentences of which the probability is constrained. Finding such model, i.e. the weights $w_1, \ldots, w_n$, with maximum entropy, but still consistent with the probabilities put on sentences, is a *convex optimisation problem* and can therefore be solved by hill-climbing algorithms.

The main disadvantage of the method is that the distribution chosen is based on the implicit maximum entropy assumption, which may yield unintended results in case parts of the distribution are heavily underspecified. Furthermore, the set of constraints can still be inconsistent, so there is no distribution to choose from in this case. Finally, the logic does not make it possible to capture general statistical first-order information in a satisfactory way.

3.3.2 *Markov Logic Networks*

A straightforward solution to the problem of inconsistent constraints is to directly consider the weights as model definition, instead of using probabilistic constraints. This approach is taken by *Markov logic networks* (MLNs) [130]. MLNs also tackle the problem of how to encode general first-order statistics, but in a quite pragmatic way. Compared to Paskin's logic, this is achieved by treating formulas with variables differently. Using a sentence such as $\forall_H\ \mathit{buglary}(H) \rightarrow \mathit{alarm}(H)$ as feature means that possible worlds were only a single burglar does not cause an alarm are as unlikely as world in which no burglar does. To solve this, in MLNs all possible groundings are considered as separate features, which however have the same weight. As we make the assumption of a finite number of objects in the domain of discourse here, the number of possible groundings is finite as well.

The problem is actually related to the expressiveness problem pointed out by Halpern [63] and Bacchus [6], discussed in Section 3.2. Instead of constraining the probability of a sentence such as $P(\forall_H\ \mathit{burglary}(H) \rightarrow \mathit{alarm}(H)) = p$, the MLN approach can be seen as constraining the probability for each single grounding of H, which could be written as $\forall_H\ P(\mathit{burglary}(H) \rightarrow \mathit{burglary}(H)) = p$. In the latter case the probability of possible worlds decreases gradually with the number of burglars causing no alarm.

A MLN therefore consists of a number of sentences $s_1, \dots, s_n$ with associated weights $w_1, \dots, w_n$ together with a finite set of objects as domain of discourse. The probability distributions over possible worlds is then defined as:

$$P(\omega) \overset{\text{def}}{=} \frac{\exp\left(\sum\limits_{i=1}^{n} w_i n_{s_i}(\omega)\right)}{\sum\limits_{\omega \in \Omega} \exp\left(\sum\limits_{i=1}^{n} w_i n_{s_i}(\omega)\right)} \tag{3.6}$$

The distribution is similar to the distribution of Paskin's approach (Equation 3.5), except that the indicator function $\mathbb{1}_s$ is replaced by the function n_s, which gives the number of true groundings of s in the possible world given as argument.

MLNs are very popular and very powerful in particular in case the number of objects is fixed and weights are learned from data. It is also considered as an advantage that new weighted sentences can just be added to an existing *knowledge base* and the knowledge base remains consistent in each case. There are however some severe drawbacks. In case the number of objects changes, learned weights are not valid any more; the predictions can in fact be changed arbitrarily by adding irrelevant objects [103, 75]. Another problem is that MLNs are not very interpretable and therefore less suited as knowledge representation language. The probability of a sentence can in general not be derived from its weight, but may depend also on the weights of other sentences. The issues are illustrated by the following example.

Example 3.4

We consider the burglary example again and start with a simple propositional model. If we want to model the fact that the probability for a burglar is 0.01, we have to assign the weight of $\ln(0.01)$ to the sentence *burglary*. However, the probability of *burglary* is changed in case more sentences are added to the theory. Suppose we for example add the sentence *burglary* $\rightarrow$ *alarm* with weight 1.0 and compute the weights of the four possible worlds:

$$\begin{aligned}
\mathit{burglary}, \quad \mathit{alarm}&\colon v_1 = \exp\left(\ln(0.01) + 1.0\right) \\
\neg\mathit{burglary}, \quad \mathit{alarm}&\colon v_2 = \exp\left(1.0\right) \\
\mathit{burglary}, \neg\mathit{alarm}&\colon v_3 = \exp\left(\ln(0.01)\right) \\
\neg\mathit{burglary}, \neg\mathit{alarm}&\colon v_4 = \exp\left(1.0\right)
\end{aligned}$$

From this we can compute the probability of burglary as:

$$P(\mathit{burglary}) = \frac{v_1 + v_3}{v_1 + v_2 + v_3 + v_4} \approx 0.0068$$

So the probability whether there is a burglar cannot be derived directly from a single weight, but one has to consider the combination of all related weights. In case we would add the sentence *alarm*, the probability of *burglary* also depends on the weight on that sentence, even though the relation between those sentences can only be seen by considering the implication. In realistic models the relations are usually much more complex. This makes MLN models hard to interpret.

Consider the following first-order version of the burglary model:

$$\begin{aligned} -4.6&: \mathit{burglary}(H) \\ 1.0&: \mathit{burglary}(H) \rightarrow \mathit{alarm}(H) \\ 1.0&: \mathit{earthquake} \rightarrow \mathit{alarm}(H) \end{aligned}$$

Such model only has a meaning if we fix the the number of houses. In case we consider a single house only, the probability of burglary is about 0.00738, in case we consider ten houses it is about 0.00686, in case we consider twenty houses it is about 0.00683 and so on. This illustrates that in general MLN models are only valid for a fixed number of objects and generalise badly.

3.3.3 *Continuous Distributions*

The concept of maximum entropy distributions can in principle also be applied to continuous distributions by replacing the sum by integration in Equation 3.4. The challenge is however how to define the features of a log-linear model, which is required to make finding maximum entropy distributions practical. Binary features do not carry enough information about continuous distributions. In fact the probability that a continuous function takes a point value, expressed as statement $\mathit{earthquake} = x$, has always probability 0.0 in case *earthquake* is distributed according to any common continuous PDF. Actually, with the statement above, one intuitively wants to express that the probability decreases with increasing difference between *earthquake* and x. Yu and et al. [160] suggest using continuous features, which however makes it hard to find the maximum entropy distribution.

An extension of Markov logic has been proposed [156], which makes use of numeric features, additional to binary ones. The authors propose to interpret $a = b$ as $-(a-b)^2$, which introduces a Gaussian penalty for the difference between a and b. The authors also propose possible translations of inequalities. Those translations are more or less arbitrary and make implicit assumptions about the underlying distribution. The general concept of arbitrary numeric features is very expressive, but makes theories hard to interpret.

Example 3.5

Consider the following hybrid MLN:

$$w\colon\ \mathit{earthquake} = 0.5$$

Which is actually sugar for:

$$w\colon\ -(\mathit{earthquake} - 0.5)^2$$

This makes *earthquake* normally distributed with mean 0.5 and a standard deviation dependent on the weight ($1/\sqrt{2w}$). As for discrete MLNs, in case the theory includes more sentences, the distribution is affected by the other sentences and the number of objects.

3.4 FIRST-ORDER BAYESIAN NETWORKS

A commonly used method to define precise probability distributions, is to define it in independent pieces instead of defining the entire distribution at the same time. Concretely, probability measures are usually defined by defining local (conditional) distributions together with an encoding of assumptions about the measure's structure, usually in terms of (conditional) independencies. For instance, a BN fixes (conditional) probabilities of a number of random variables. This, together with the graph's structure, fixes the probability of all events and therefore defines a single probability measure. Here, we discuss ways to also achieve this similarly with logic.

3.4.1 *Probabilistic Functional Programming*

A way to ensure a unique distribution is to use functions only. If one has given definitions for a number of dependent functions, the value of each function application is uniquely determined. Here that functions are dependent means, that function symbols of some functions can be used in the definition of other ones. This is the same principle as used by ordinary functional programming languages. If we assume an independent probabilistic choice for the definition of each function, this defines a single joint probability distribution, similarly to a BN. As in BNs, the distribution is only well defined in case there are no cyclic dependencies.

Unlike LP, functional programming can be seen is a subset of FOL, although practical languages also come with facilities to help writing actual programs, as type systems and ways to define data structures. Also functions are required to be unambiguously defined to eliminate the indeterminateness immanent to FOL, which is in actual languages enforced by providing special syntax to define functions, such as case statements. Probabilistic functional languages furthermore make use of functions yielding probability distributions instead of probability distributions over functions, providing a more compact and intuitive representation, which is however equally expressive. There are several languages based on this idea, such as Church [57], BLOG [102] and Figaro [120] though not all of them realise the concept in a pure way.

Example 3.6 ---

In the example we use syntax which does not directly correspond to any actual language, but is representative for this family of languages. The probability distribution of the function *alarm* could be defined by functions yielding distributions as:

$$\begin{aligned} \mathit{burglary} &= \{0.01\colon \mathit{true}, 0.99\colon \mathit{false}\} \\ \mathit{alarm} &= \begin{cases} \{0.99\colon \mathit{true}, 0.01\colon \mathit{false}\} & \text{if } \mathit{burglary} = \mathit{true} \\ \{0.02\colon \mathit{true}, 0.98\colon \mathit{false}\} & \text{if } \mathit{burglary} = \mathit{false} \end{cases} \end{aligned}$$

Both distributions above have to be seen as independent. The probability of $\mathit{alarm} = \mathit{true}$ is $0.01 \cdot 0.99 + 0.99 \cdot 0.02 = 0.0297$.

Formally, suppose we have n function labels $f_1, \ldots, f_n$. For the i-th function there are m_i possible definitions $\mathbf{D}_i = \{d_{i1}, \ldots, d_{im_i}\}$, with associated probabilities $p_{i1}, \ldots, p_{im_i}$. We can define an event, equivalent to a sentence s, as a special case of Equation 3.1:

$$e_s \overset{\text{def}}{=} \{d_1 \in \mathbf{D}_1, \ldots, d_n \in \mathbf{D}_n \mid f_1 \equiv d_1 \wedge \cdots \wedge f_n \equiv d_n \models s\} \tag{3.7}$$

The sentence s here can be some statement about the functions, e.g. $f(v) = w$ and $f(v) = g(v)$. We use the notation $f \equiv d$ to indicate that the function with label f obeys definition d. The probability of one assignment of function definitions is the product of the associated probabilities, as we assume the choices of definitions to be independent:

$$P(f_1 \equiv d_1 \wedge \cdots \wedge f_n \equiv d_n) \overset{\text{def}}{=} p_1 \cdot \cdots \cdot p_n \tag{3.8}$$

Here each p_i is the probability associated with definition d_i. The probability of an event is then the sum of all those probabilities:

$$P(e_s) \overset{\text{def}}{=} \sum_{\omega \in e_s} P(\omega) \tag{3.9}$$

First-order models can be defined by using variables in function definitions, as is commonly done in all programming languages. The variables are implicitly universally quantified, which actually expresses that the same definition holds for all values of the variables. So in contrast to MLNs, the model generalises to an arbitrary number of objects. Statistical information however cannot straightforwardly be expressed in this way, as discussed in Section 3.2. However, given the guarantee that functions are defined unambiguously, provided by functional languages, it is not possible to add inconsistent knowledge. Practically, observations about particular objects are treated as observations updating the distribution defined by the model.

Example 3.7

The function alarm can be defined for multiple houses as:

$$\mathit{alarm}(H) = \begin{cases} \{0.99\colon \mathit{true}, 0.01\colon \mathit{false}\} & \text{if } \mathit{burglary}(H) = \mathit{true} \\ \{0.02\colon \mathit{true}, 0.98\colon \mathit{false}\} & \text{if } \mathit{burglary}(H) = \mathit{false} \end{cases}$$

The variable H has to be considered as being implicitly universally quantified. This definition then actually means that the probability for an alarm depending on burglary is the same for all houses. As discussed before, this does not properly encode statistical information about randomly selected houses, but observations about particular houses can be used to update the distribution over all houses defined above.

3.4.2 *Probabilistic Logic Programming*

LP provides another way to avoid indeterminateness. As functional programming guarantees that each function application has a unique value, a LP theory unambiguously defines which predicates are true and false (this is the basic assumption we make in Section 2.1.3). So LP can also be taken as a basis for precise probability distributions and has compared to the functional approach the advantage that it is for some domains more suited as knowledge representation tool. Probabilistic LP is well studied and there are a large number of concrete languages and implementations, for instance ICL [122], *PRISM* [138], ProbLog [125] and *causal probabilistic logic* (CP-logic) [152]. We introduce Sato's *distribution semantics* [137], which is the most basic probabilistic LP semantics with a rigorous development for an infinite number of objects in the domain of discourse. Concrete languages have a number of extensions to support model construction, which are not discussed here in detail (see [40] for a detailed discussion of differences of concrete languages).

The purpose of the distribution semantics is to extend the semantics of LP towards a probabilistic semantics by guaranteeing the existence of a unique probability distribution consistent with Kolmogorov's probability axioms. The key property here is the fact that logic programs always have a unique model. Sato uses the traditional least model semantics, but the concept can be generalised to all programs for which some kind of unique model can be defined. Suppose we split program $\mathbf{L}$ into rules $\mathbf{R}$ and facts $\mathbf{F}$, i.e. $\mathbf{L} = \mathbf{F} \cup \mathbf{R}$. Facts are rules with an empty body and we assume that facts never occur as the head of a rule. The semantics assume a probability measure on facts, so facts are considered as being binary random variables, taking values *true* or *false*. This distribution can uniquely be extended to all predicates defined by rules $\mathbf{R}$. Concrete languages usually require to define the measure on facts by independent probabilities, however without loss of generality, as dependencies can be introduced by the structure of the rules and additional auxiliary facts. This concept is illustrated in Figure 3.2. A grounded program, i.e. a program in which each variable is replaced by all possible ground terms, potentially includes an infinite, but countable, number of ground facts. For instance, a probabilistic process could include random variables representing whether it rains on particular days (a fact) for an infinite number of future days. First-order models can therefore be defined in the same way as in the functional approach discussed above: one can state prior probabilities for all possible objects, but not properly state statistical information about randomly selected objects.

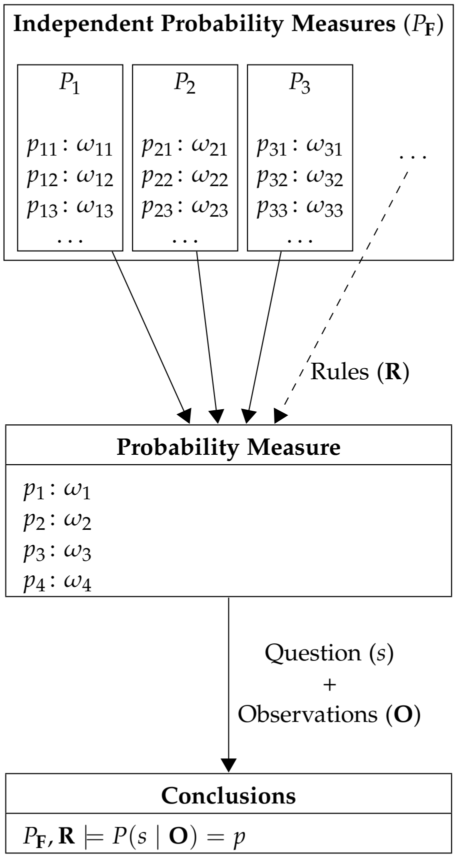


Figure 3.2: Distribution Semantics Illustration

Associated to the distribution on the facts we have a sample space Ω_F, where each element is a potentially infinite sequence of Boolean values indicating the truth value of all facts. The truth value of each literal in **L** is determined by such a truth assignment and we can therefore speak of the model associated with an element $\omega_{\mathbf{F}} \in \Omega_F$ and denote it by $M_{\mathbf{L}}(\omega_{\mathbf{F}})$. Given a probability measure $P_{\mathbf{F}}$ defined on the event space, which is the powerset of Ω_F, it is possible to extend this distribution to all *atoms* occurring in the program in a unique way. We refer to the resulting distribution as $P_{\mathbf{L}}$. The used sample space $\Omega_{\mathbf{L}}$ is defined similarly as the event space of the facts, but includes all atoms. The event space is again the sample space's powerset. We show formally how a probability measure on a potentially infinitely large event space can properly be defined in detail, as the work in the thesis makes use of the same idea. The construction is based on finite measures restricted to the first n atoms, which can then be extended to a measure on the infinite event space using the extension theorems of probability measures [110, III.3]. A finite measure on the first n atoms is defined as:

$$P^n_{\mathbf{L}}(\omega^n_{\mathbf{L}}) \overset{\text{def}}{=} P_{\mathbf{F}}\big(\{\omega_{\mathbf{F}} \in \Omega_F \mid M_{\mathbf{L}}(\omega_{\mathbf{F}}) \models \omega^n_{\mathbf{L}}\}\big) \tag{3.10}$$

Here we use an element $\omega_{\mathbf{L}}$ of the sample space as a logical formula representing a conjunction that determines for each atom whether it is true or false. The event of which the probability is computed here is similar to the one in Equation 3.1, except that only the fact-part of the sample space is considered. For each $P^n_{\mathbf{L}}$ defined like this, $P^{n+1}_{\mathbf{L}}$ is *compatible*, which means:

$$\sum_{a \in \{true, false\}} P^{n+1}_{\mathbf{L}}(\omega^n_{\mathbf{L}}, a) = P^n_{\mathbf{L}}(\omega^n_{\mathbf{L}}) \tag{3.11}$$

Therefore, the finite probability measures construct a probability measure for the infinite number of atoms.

The probability measure defined on all atoms makes it possible to compute the probability of any arbitrary query sentence q by $P_{\mathbf{L}}\big(\{\omega_{\mathbf{L}} \in \Omega_{\mathbf{L}} \mid \omega_{\mathbf{L}} \models q\}\big)$. Probabilities of queries with finite grounding can be computed exactly, since finite measures can be used then.

Example 3.8

We build the burglary model starting with independent probabilities on binary facts:

$$\begin{aligned} P(burglary) &= 0.01 \\ P(alarm\_if\_burglary) &= 0.99 \\ P(alarm\_if\_no\_burglary) &= 0.02 \end{aligned}$$

We can then define the predicate *alarm* as:

$$\begin{aligned} &alarm \leftarrow alarm\_if\_burglary, \quad burglary \\ &alarm \leftarrow alarm\_if\_no\_burglary, not(burglary) \end{aligned}$$

Table 3.3 shows all elements of the sample space on F, together with models and probabilities associated with the elements. We can now compute, for instance, the probability of *alarm* by summing over all cases in which *alarm* is included in the model: $P(alarm) = 0.000198 + 0.019602 + 0.009702 + 0.000198 = 0.0297$. In the same way we can compute the probability of other logical sentences. Examples are $P(alarm \land burglary) = 0.009702 + 0.000198 = 0.0099$ and $P(alarm \lor burglary) = 0.000198 + 0.019602 + 0.000098 + 0.000002 + 0.009702 + 0.000198 = 0.0298$.

Finally, we also give a first-order version of the model:

$$\begin{aligned} &\forall_H\, P\big(burglary(H)\big) &&= 0.01 \\ &\forall_H\, P\big(alarm\_if\_burglary(H)\big) &&= 0.99 \\ &\forall_H\, P\big(alarm\_if\_no\_burglary(H)\big) &&= 0.02 \end{aligned}$$

$\omega_{\mathbf{F}} =$ $(burglary,$ $alarm\_if\_burglary,$ $alarm\_if\_no\_burglary)$	$M_{\mathbf{L}}(\omega_{\mathbf{F}})$	$P_{\mathbf{L}}(\omega_{\mathbf{F}})$
$(false, false, false)$	$\{\}$	0.009702
$(false, false, true)$	$\{alarm\_if\_no\_burglary, alarm\}$	0.000198
$(false, true, false)$	$\{alarm\_if\_burglary\}$	0.960498
$(false, true, true)$	$\{alarm\_if\_burglary, alarm\_if\_no\_burglary, alarm\}$	0.019602
$(true, false, false)$	$\{burglary\}$	0.000098
$(true, false, true)$	$\{burglary, alarm\_if\_no\_burglary\}$	0.000002
$(true, true, false)$	$\{burglary, alarm\_if\_burglary, alarm\}$	0.009702
$(true, true, true)$	$\{burglary, alarm\_if\_burglary, alarm\_if\_no\_burglary, alarm\}$	0.000198

Table 3.3: Distribution Semantics Example

$$\begin{aligned} &alarm(H) \leftarrow alarm\_if\_burglary(H), \quad burglary(H) \\ &alarm(H) \leftarrow alarm\_if\_no\_burglary(H),\ not(burglary(H)) \end{aligned}$$

In the rules, the variable H is universally quantified, which is implicit in LP rules. One could define a rule representing that there is an alarm in a house and all next houses:

$$all(H) \leftarrow alarm(H),\ all(next(H))$$

This defines a probability distribution for all houses $a, next(a), next(next(a)), \ldots$.

3.4.3 *Continuous Distributions*

The directed way of defining distributions, can also be extended to continuous distributions. Distributions can be defined by PDFs or conditional PDFs, meaning that the parameters of the PDFs depend on the value of other random variables. As an example, we briefly discuss *distributional clauses* (DC) [61], which builds upon a generalisation of the distribution semantics. The language explicitly introduces random variables, to conveniently represent distributions with multiple states, which are potentially continuous. Inference is not possible in an exact way and is therefore based on sampling.

Example 3.9

Recall the deterministic burglary program from Example 2.9:

$$\begin{aligned}&alarm(H) \leftarrow burglary(H)\\&alarm(H) \leftarrow earthquake(E),\ \langle E > 5.5\rangle\end{aligned}$$

With DC we can model the problem in a very similar way, except that prior distributions are required:

$$\begin{aligned}burglary(H) &\sim [0.01\colon yes, 0.99\colon yes]\\earthquake &\sim gamma(1.0, 2.0)\end{aligned}$$

$$\begin{aligned}alarm(H) &\leftarrow \simeq burglary(H) = yes\\alarm(H) &\leftarrow \simeq earthquake > 5.5\end{aligned}$$

Here $burglary(H)$ is a random variable with two possible states. In the rule it is used together with $\simeq$, which maps a random variable to its value. Continuous random variables are used in the same way, such as *earthquake* in the example. The distribution is in this case defined by a commonly used PDF.

An example of a conditional continuous distributions, is a measure of the earthquake's strength, which depends on the actual strength, but introduces some (Gaussian) error:

$$measure \sim gamma(\simeq earthquake, 1.0)$$

All languages allowing to define precise and continuous distributions basically share the same drawbacks. Firstly, one is forced to choose an appropriate PDF and parameters to describe the distribution, which usually requires data, as it is even harder for humans to estimate than discrete probabilities. The main technical drawback is that *marginal* probabilities are usually not computable, but can only be approximated.

3.5 DISCUSSION

In this chapter we discussed and compared several ways to combine logics with probability theory. The main insight is, that making such combination practical requires balancing pragmatics with expressivity. An intuitive representation of probability distributions and preventing inconsistencies is only possible by forcing a very structured way of assigning probabilities to logical statements.

Languages providing such structure are actually similar to BNs, with the difference that they are first-order. One restriction is however that such language cannot properly encode general statistical knowledge, which can however pragmatically be solved by viewing a probability on a first-order statements as independent probabilities on all possible groundings and allow updating knowledge by conditioning on evidence only. Another restriction, shared with all formalisms defining a unique probability distribution, is that, as soon as ranges of variables become infinitely large, probabilities cannot be computed exactly any more, but one has to rely on approximation methods, not providing any rational guarantee on the quality of results. Logics in general however often allow to draw rational conclusions, i.e. they provide *sound* inference, also for the case that infinite ranges are involved. We will further deal with this issue in the thesis.

4 A NEW PROBABILISTIC CONSTRAINT LOGIC

In this chapter we introduce the novel language *Probabilistic Constraint Logic Programming* (PCLP).

4.1 IN FAVOUR OF IMPRECISE PROBABILITIES

In contrast to most recent *probabilistic logic* languages, the work of this thesis considers *imprecise* rather then precise probability distributions. As discussed, imprecise probabilities are very useful in case no precise probabilities are known. The main reason to use imprecise probabilities in this work is however that some statements about probabilities on an infinite number of *possible worlds*, which cannot be decided in a precise setting, can be in an imprecise setting. As an example consider the statement $P(a < 1) = 0.1 \wedge P(a > 1) = 0.9$. From this one can conclude $P(a > 0) \geq 0.9$ and $P(a < 0) \leq 0.1$, without having to resort to approximation methods as sampling.

While employing imprecise probabilities, we want to avoid the disadvantages of other imprecise methods. First of all, we want that our language remains decidable (at least if considering a finite number of *random variables*), in contrast to the most general probabilistic logics, but also in contrast to defining a point distribution on continuous variables by means of *probability density functions* (PDFs). Also, we want the guarantee that definitions are always consistent, as provided by most recent languages, but unlike defining *credal sets* by putting arbitrary probabilistic constraints on sentences. Finally, we desire that the *inference* complexity compared to precise probabilistic reasoning is increased as less as possible. Concretely, inference should be in the same complexity class, which is for instance not the case for *locally defined credal networks* (LDCNs).

We introduce a novel language, PCLP, which possesses all those properties. The inference complexity is compared to existing methods in Figure 4.1. Here inference complexity increases from left to right. On the right side there are undecidable methods. On the left side we have precise distributions. PCLP is placed just right of precise distributions. It is more expressive, as it supports certain kinds of imprecise reasoning, but remains in the same complexity class. It is nevertheless more complex in terms of complexity parametrised by the problem structure (Section 5.3.2). Decidable imprecise formalisms, as LDCNs, are more expressive, at the price of increased inference complexity.

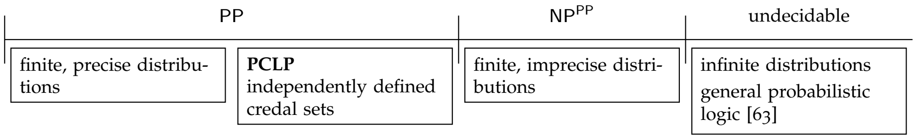

PP
NP^PP
undecidable
finite, precise distributions
PCLP
independently defined credal sets
finite, imprecise distributions
infinite distributions
general probabilistic logic [63]

Figure 4.1: Complexity of Probabilistic Reasoning Methods

4.2 ASSIGNING PROBABILITY MASS TO EVENTS

We first give an overview of the basic idea PCLP is based on, before we introduce the language formally. Credal sets in PCLP are defined by splitting the entire probability mass of 1.0 into a finite number of pieces and assign it to *events*. To assign probability mass to an event means, that the probability mass is somehow distributed over that event.

Example 4.1

Assume we have a single random variable with the real numbers as range and assign some probability mass to the set of all values between 1 and 3. Figure 4.2 depicts some possible ways of how the probability mass can be distributed: it can be distributed uniformly over the entire set (Figure 4.2a) or only parts of it (Figures 4.2b and 4.2c) or distributed in a more complex manner (Figure 4.2d). These are just a few examples; there are actually uncountably many ways to distribute the probability mass over that set.

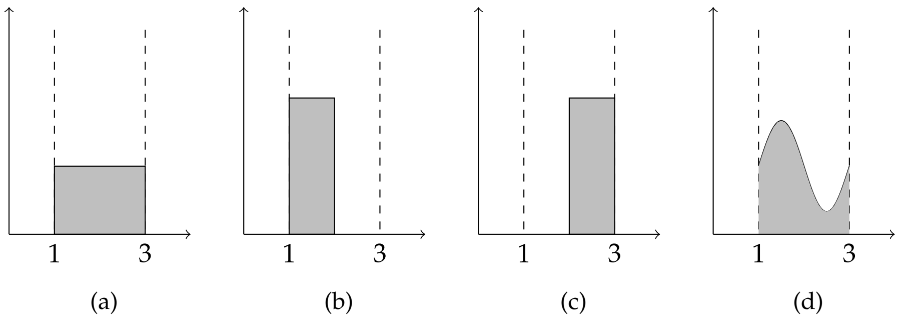


Figure 4.2: Examples of Possible Distributions of Probability Mass over all Real Values Between 1 and 3

As there are multiple possible ways in which the probability mass can be distributed, this implies that this does not define a unique probability distribution, but a credal set. The credal set is however never empty, which is a consequence of splitting the probability mass, instead of assigning arbitrary probabilities to events. Furthermore, in case the number of probability assignments is finite, computing probabilistic bounds on events is decidable.

Example 4.2

Consider the following constraint logic program:

$$q \leftarrow \langle \mathbf{V}_1 \geq 0 \rangle$$

Suppose we define the probability distribution of $\mathbf{V}_1$ by assigning a probability mass of 0.1 to the set of all values smaller than -1 (Set 1), 0.3 to the closed interval $[-1, 1]$ (Set 2) and 0.6 to the set of all values larger than 1 (Set 3). It is clear that no matter how the probability mass is distributed over the values in Set 1 the probability that q can be derived is always 0.0. For Set 2 the probability could be 0.0 in case all probability mass is distributed to values below 0.0, or 0.3 in the opposite case. For Set 3 the query can always be derived, no matter how the probability mass is distributed. We conclude that the probability of q is at least 0.6 and at most 0.9.

For encoding the structure of problems, we use the idea of the *distribution semantics*. *Independent* credal sets are combined by deterministic rules, as shown in Figure 4.3. This makes inference as complex as precise probabilistic inference. In contrast to defining independent credal sets, for instance LDCNs define conditional credal sets independently. However, to handle those conditional dependencies during inference, it has to be decided which extreme probability points lead to extreme probabilities for other random variables, which cannot be done independently. This makes inference $\mathsf{NP}^{\mathsf{PP}}$-complete. In contrast, inference for PCLP remains in PP, as will be proven later in this thesis (Section 5.3.1).

4.3 MOTIVATING EXAMPLES

To illustrate the expressive power of the new language PCLP, a few typical examples are presented.

4.3.1 *Fruit Selling*

We start with an example concerning the likelihood that consumers will buy a certain kind of fruit, based on [11]. Since we have a first-order formalism, this generalises easily to an arbitrary number of kinds (in the example: apples and bananas). The main goal is to show how PCLP can deal with continuous distributions.

Yield of fruit is clearly relevant for its price. We model the yield of fruit with normally distributed random variables (denoted by a string starting with an upper case letter and in bold):

$$\begin{aligned} \mathbf{Yield}(\mathit{apple}) &\sim \mathcal{N}(12\,000.0, 1000.0) \\ \mathbf{Yield}(\mathit{banana}) &\sim \mathcal{N}(10\,000.0, 1500.0) \end{aligned}$$

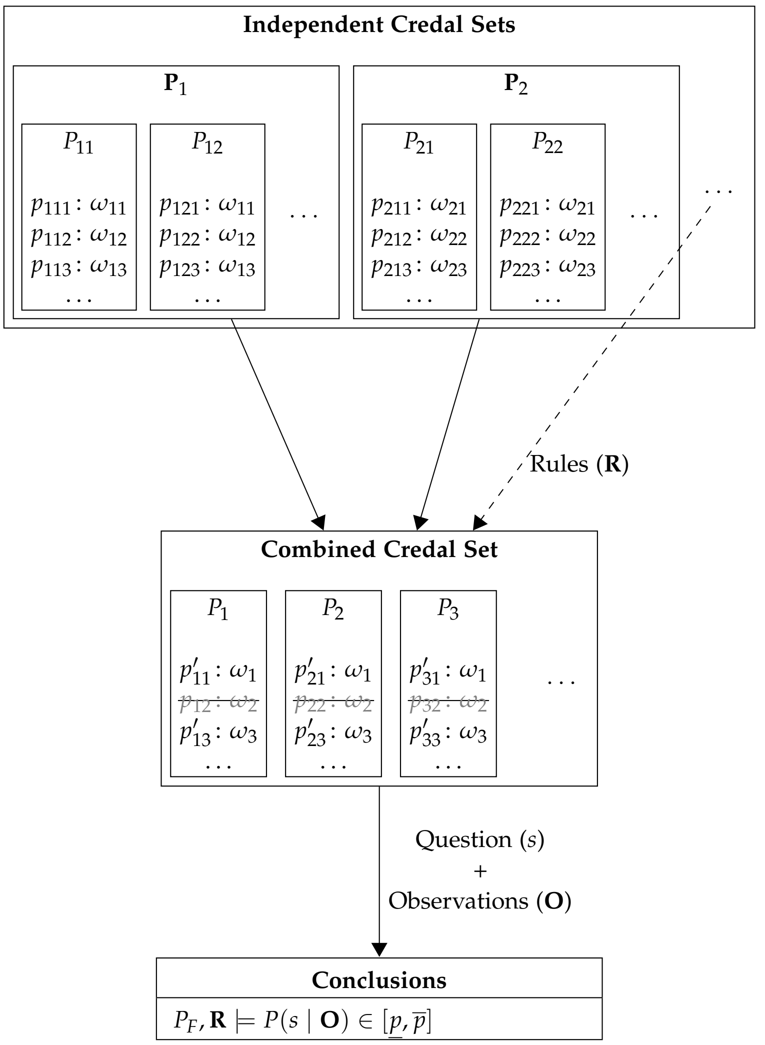


Figure 4.3: PCLP Semantics Illustration

The price is also influenced by government support, which is modelled by discrete random variables:

$$\begin{aligned}
&\mathbf{Support}(apple) &&\sim \{0.3\colon yes, 0.7\colon no\}\\
&\mathbf{Support}(banana) &&\sim \{0.5\colon yes, 0.5\colon no\}
\end{aligned}$$

The basic price linearly depends on the yield, which is expressed as a deterministic logic fact:

$$\begin{aligned}
&basic\_price(apple, &&250 - 0.007 \cdot \mathbf{Yield}(apple))\\
&basic\_price(banana, &&200 - 0.006 \cdot \mathbf{Yield}(banana))
\end{aligned}$$

In case the price is supported it is raised by a fixed amount:

$$\begin{aligned}
&price(Fruit, BPrice + 50) \leftarrow basic\_price(Fruit, BPrice),\ \langle \mathbf{Support}(Fruit) = yes\rangle \\
&price(Fruit, BPrice) \qquad\quad\ \leftarrow basic\_price(Fruit, BPrice),\ \langle \mathbf{Support}(Fruit) = no\rangle
\end{aligned}$$

Fruit is a *logical* variable (not denoted in bold) which can take kinds of fruit, e.g. *apple* and *banana*, as possible instantiations. Here we make use of the special *predicate* $\langle\rangle$, which represents probabilistic events; for example, $\langle \mathbf{Support}(Fruit) = yes\rangle$ is true in case the random variable $\mathbf{Support}(Fruit)$ takes the value *yes*. In *constraint logic programming* (CLP) [72] a similar predicate is used to represent constraints.

At which maximum price customers still buy a certain fruit is modelled by a gamma distribution:

$$\begin{aligned}
\mathbf{Max\_price}(apple) &\sim \Gamma(10.0, 18.0) \\
\mathbf{Max\_price}(banana) &\sim \Gamma(12.0, 10.0)
\end{aligned}$$

Thus, a customer is willing to buy in case the price is equal to or less than the maximum price, which can be expressed by the following first-order rule:

$$buy(Fruit) \leftarrow price(Fruit, P),\ \langle P \leq \mathbf{Max\_price}(Fruit)\rangle$$

The interesting question to ask given this *knowledge base* is whether customers buy a certain fruit. As it is uncertain which of the events, specified by the occurrences of $\langle\rangle$, actually occur, the only answer we can give is how likely such statements are, e.g. $P(buy(apple))$ or $P(buy(apple) \vee buy(banana))$. Another possible question is what the probability is that customers buy apples given that we know what the least maximum yield will be, e.g. $P(buy(apple) \mid \langle \mathbf{Yield}(apple) \geq 10\,000.0\rangle)$.

Note that such probabilities cannot be computed exactly, as they require computing probabilities of linear inequalities between different kinds of continuous distributions, which is in general not computable. The work of this thesis however, allows one to determine approximations with known maximum errors, for example:

$$\begin{aligned}
&P\big(buy(apple)\big) &&\approx 0.464 \pm 0.031 \\
&P\big(buy(banana)\big) &&\approx 0.162 \pm 0.031 \\
&P\big(buy(apple) \vee buy(banana)\big) &&\approx 0.552 \pm 0.054
\end{aligned}$$

The maximum error is determined by the used approximation scheme and can be made arbitrarily small, as explained in Section 4.5.2.5.

4.3.2 *Diabetes Mellitus*

The next, medical example shows how PCLP can be used to model problems involving continuous distributions as well as imprecise probabilities. The latter means one uses bounds rather than precise probability estimates, which is a way to handle situations concerned with insufficient data to reliably estimate probabilities. Such situations frequently occur in clinical research. A possible approach in such cases is to express uncertainty about probabilities by yet another probability distribution, i.e. using second-order probability distributions. In contrast, the approach of imprecise probabilities assumes that all possible probabilities within the specified range are possible, but furthermore expresses complete ignorance of what the actual probability is. So imprecise probabilities relieve from specifying a second-order distribution, which requires knowledge or an amount of data not always available, at the price of making a hard choice of which probabilities are possible and which are not.

The example concerns diabetes mellitus type 2, which is a complex disorder in which several metabolic control mechanisms are disturbed. A first step in its treatment is to regulate the glucose metabolism. In diabetic patients glucose, although present in abundance in the extracellular fluid with the exception of the cerebrospinal fluid, is unable to cross the cellular membrane and cells therefore lack their usual energy resource (often called "starvation amidst abundance"). A standard test to check the quality of glucose control is the measurement of fasting blood glucose levels. Furthermore, the levels of glycated hemoglobin (HbA1c) offer insight into the effectiveness of long-term (8 to 12 weeks) glucose control. Clearly, the fasting blood glucose and HbA1c measurements are related, although only on average.

While type 2 diabetes is mostly related to lifestyle-related factors, recent biomedical research indicates that various genetic factors play a role in its onset. In [148], familial risk of type 2 diabetes was classified as average, moderate, or high. In the US population, 69.8% were in the average, 22.7% in the moderate, and 7.5% in the high familial risk group. In PCLP this can be represented as follows:

$$\textbf{Predisposition} \sim \{0.698\colon \textit{average}, 0.227\colon \textit{moderate}, 0.075\colon \textit{high}\}$$

According to [148], the crude prevalences of diabetes within each risk category was between 5.4% and 6.6% in the average risk group, between 13.1% and 16.7% in the moderate risk group, and between 26.6% and 33.6% in the high risk group.

$$\begin{aligned}
&\textbf{DM\_if\_AverageRisk} &&\sim \{0.054\colon \textit{yes}, 0.934\colon \textit{no}\}\\
&\textbf{DM\_if\_ModerateRisk} &&\sim \{0.131\colon \textit{yes}, 0.833\colon \textit{no}\}\\
&\textbf{DM\_if\_HighRisk} &&\sim \{0.266\colon \textit{yes}, 0.664\colon \textit{no}\}
\end{aligned}$$

The imprecision in these conditional probabilities is encoded by leaving part of the probability mass unspecified; e.g. for the high risk group, at least 0.266 of the probability mass is in the *yes* state, 0.664 is in the *no* state, and the remainder is unspecified.

An *atom dm* can be defined to indicate whether a patient suffers from diabetes by defining logical *clauses* for each case of **Predisposition**:

$$\begin{aligned}
&dm \leftarrow \langle \mathbf{DM\_if\_AverageRisk} = \mathit{yes}\rangle, && \langle \mathbf{Predisposition} = \mathit{average}\rangle\\
&dm \leftarrow \langle \mathbf{DM\_if\_ModerateRisk} = \mathit{yes}\rangle, && \langle \mathbf{Predisposition} = \mathit{moderate}\rangle\\
&dm \leftarrow \langle \mathbf{DM\_if\_HighRisk} = \mathit{yes}\rangle, && \langle \mathbf{Predisposition} = \mathit{high}\rangle
\end{aligned}$$

We can now compute, for example, a probability range for the probability of diabetes which yields: $0.087379 \leq P(dm) \leq 0.109177$.

To illustrate a combination with continuous variables, suppose the level of glucose is represented by two normal distributions $\mathcal{N}(\mu, \sigma^2)$, where μ and σ denote the mean and standard deviation for the cases where the patient either has or does not have diabetes:

$$\begin{aligned}
&\mathbf{Gluc\_if\_DM} && \sim \mathcal{N}(7.5, 3.8)\\
&\mathbf{Gluc\_if\_not\_DM} && \sim \mathcal{N}(5.79, 0.98)
\end{aligned}$$

Another continuous variable can be used to represent the level of HbA1c, which we assume to linearly depend on the level of glucose plus some noise which depends on whether the patient is a diabetic. The noise variables can be modelled by two random variables, after which HbA1c can be defined:

$$\begin{aligned}
&\mathbf{Noise\_if\_DM} && \sim \mathcal{N}(0.0, 3.3)\\
&\mathbf{Noise\_if\_not\_DM} && \sim \mathcal{N}(0.0, 0.3)
\end{aligned}$$

$$\begin{aligned}
&hba1c(1.4 + 0.92 \cdot \mathbf{Gluc\_if\_DM} + \mathbf{Noise\_if\_DM}) && \leftarrow dm\\
&hba1c(0.6 + 0.9 \cdot \mathbf{Gluc\_if\_not\_DM} + \mathbf{Noise\_if\_not\_DM}) && \leftarrow not(dm)
\end{aligned}$$

Using this representation, it is possible, for instance, to compute bounds on the probability of diabetes given that the level of HbA1c is larger than 7.2. This *evidence* can be encoded using the following clause:

$$e \leftarrow hba1c(H),\ \langle H > 7.2\rangle$$

The following probability bounds can then be computed:

$$0.416 \leq P(dm \mid e) \leq 0.554$$

Note that the imprecision results from the fact that we use imprecise probabilities, as well as from the fact that continuous distributions are approximated. Again, a better approximation can be found by investing more computation time.

4.3.3 *Running Example: Fire on a Ship*

As a final example, we introduce a short case description, which will be used to illustrate several concepts throughout this chapter. Suppose there is a fire in a compartment of a ship. The heat causes the hull of that compartment to warp and if the fire is not extinguished within 1.25 minutes the hull will breach. After 0.75 minutes the fire will spread to the compartment behind. This means that if the fire is extinguished within 0.75 minutes the ship is saved for sure:

$$\mathit{saved} \leftarrow \langle \mathbf{Time\_Comp_1} < 0.75 \rangle$$

In the other compartment the hull will breach 0.625 minutes after the fire breaks out. In order to reach the second compartment the fire in the first one has to be extinguished. So both fires have to be extinguished within $0.75 + 0.625 = 1.375$ minutes. Additionally, the fire in the first compartment has to be extinguished within 1.25 minutes, because otherwise the hull breaches there. The second compartment is however more accessible, such that four fire-fighters can extinguish the fire at the same time, which means they can work four times faster:

$$\mathit{saved} \leftarrow \langle \mathbf{Time\_Comp_1} < 1.25 \rangle, \langle \mathbf{Time\_Comp_1} + 0.25 \cdot \mathbf{Time\_Comp_2} < 1.375 \rangle$$

Suppose further that the time durations required to extinguish the fires are exponentially distributed:

$$\mathbf{Time\_Comp_1} \sim \mathrm{Exp}(1)$$
$$\mathbf{Time\_Comp_2} \sim \mathrm{Exp}(1)$$

The interesting question here is how likely it is that the ship is saved, i.e. $P(\mathit{saved})$.

4.4 SYNTAX

PCLP can be seen as a template language, leaving the choice of which constraint theories to fill in. So we discuss the syntax in two steps: First, we discuss constraint theory agnostic rules (Section 4.4.1). Then, we give some examples of constraint theories, which can be used to form a concrete language (Section 4.4.2).

4.4.1 *Constraint Theory Agnostic Rules & Random Variable Definitions*

An overview of the PCLP syntax is given in Table 4.1. In the logical language, we use *logic programming* (LP) rules extended with constraints, similar to CLP, as discussed in Section 2.1.3.2. Elements of rule bodies may consist of both literals and constraints of the form $\langle \varphi_i \rangle$. We define credal sets by a number of *random variable definitions*. Random variable definitions define credal sets for groups of random variables, such that the definition of each random variable depends only on a finite number of other ones. Definitions have the form:

$$(\mathbf{V}_1, \ldots, \mathbf{V}_n)(X_1, \ldots, X_m) \sim \{p_1 \colon \varphi_1, \ldots, p_l \colon \varphi_l\},$$

where each $\mathbf{V}_i$ is a random variable label, X_i is a parameter, p_i is a probability and φ_i is a constraint. This intuitively means that a piece of the entire probability

Concept	**Syntax**
Random Variable Definition	$(\mathbf{V}_1, \ldots, \mathbf{V}_n)(t_1, \ldots, t_m) \sim \{p_1 \colon \varphi_1, \ldots, p_l \colon \varphi_l\}$
Rule	$h \leftarrow b_1, \ldots, b_n$
Constraint φ_i	constraint (e.g. $\mathbf{V} \in \{a, b\}$, $\mathbf{V} \leq \mathbf{W}$)
Body Element b_i	a_i, $not(a_i)$ or $\langle \varphi_i \rangle$
Atom a_i	$q(t_1, \ldots, t_n)$
Predicate q	some predicate name (starting with lower case)
Term t_i	some Prolog term (e.g. a, 3, $a(b, 3)$), may contain logical variables (e.g. $a(X)$)
Logical Variable X_i	some variable name (starting with upper case)
Random Variable $\mathbf{V}_i$	some random variable name (starting with upper case, bold)
Probability p_i	number in the range $[0.0, 1.0]$

Table 4.1: PCLP Syntax Overview

mass p_i is assigned somehow to the event represented by φ_i, following the idea discussed previously.

The random variable definitions define random variables

$$\mathbf{V}_1(X_1, \ldots, X_m), \ldots, \mathbf{V}_n(X_1, \ldots, X_m),$$

for all groundings of $X_1, \ldots, X_m$. All labels have the same parameters to make sure all X_i are ground in case a single ground instance of one of the defined random variables is used in the program. There must be at least one random variable label in the list and all labels must be distinct. If there is only one, the brackets may be left out.

Example 4.3

A single random variable, representing the temperature, could for instance be defined as:

$$\mathbf{Temperature} \sim \ldots$$

We can also define random variables representing the temperature of all infinitely many future days as:

$$\mathbf{Temperature}(Day) \sim \ldots$$

We can finally define the temperature and humidity together, which makes sense in case they are modelled by a multivariate distribution, i.e. the dependency between the random variables cannot be expressed by the logical structure:

$$(\mathbf{Temperature}, \mathbf{Humidity})(\mathit{Day}) \sim \dots$$

In case multiple definitions concern the same random variable, the definition occurring first in the program defines the variable. Labels occurring together in a definition can only occur in another definition in an identical list of labels, to make sure all random variables are always defined unambiguously.

Example 4.4

Suppose we want to specify distributions for the temperature on future days, but want to make an exception for Day 0. We could do this as follows:

$$\begin{aligned}\mathbf{Temperature}(0) &\sim \dots\\ \mathbf{Temperature}(\mathit{Day}) &\sim \dots\end{aligned}$$

The second line would define $\mathbf{Temperature}(0)$ as well, but it is not used in this case because the special definition for Day 0 occurs first.

The following definition is invalid:

$$\begin{aligned}\mathbf{Temperature}(\mathit{Day}) &\sim \dots\\ (\mathbf{Temperature}, \mathbf{Humidity})(\mathit{Day}) &\sim \dots\end{aligned}$$

Temperature is already defined by the first line; the definition in the second line may contradict that definition.

A random variable definition

$$(\mathbf{V}_1, \dots, \mathbf{V}_n)(X_1, \dots, X_m) \sim \{p_1 \colon \varphi_1, \dots, p_l \colon \varphi_l\},$$

defines an n-dimensional credal set on:

$$\mathbf{V}_1(X_1, \dots, X_m), \dots, \mathbf{V}_n(X_1, \dots, X_m)$$

Each p_i is a real number between 0 and 1 and $p_1 + \cdots + p_l = 1$. The φ_i are constraint definitions using the $\mathbf{V}_i$ as placeholder for the random variables. Each φ_i must be consistent, to make sure all probability mass is assigned to non-empty events. The p_i and φ_i can be expressions including $X_1, \dots, X_m$.

Example 4.5

Consider the following valid definition:

$$\mathbf{Temp}(\mathit{Day}) \sim \{0.2 \colon \mathbf{Temp} < 0, 0.8 \colon \mathbf{Temp} > 0\}$$

The definition could also depend on the value of the logical variables *Day*:

$$\mathbf{Temp}(\mathit{Day}) \sim \{0.2 \colon \mathbf{Temp} < \mathit{Day}/1000, 0.8 \colon \mathbf{Temp} > \mathit{Day}/1000\}$$

The definition is only valid if groundings for *Day* are restricted to numbers. It actually expresses that the temperature in the future will increase on average.

Note that in the definitions in the examples (Section 4.3) we use syntactic sugar. For example consider this definition from the examples:

$$\mathbf{DM\_if\_HighRisk} \sim \{0.266\colon \mathit{yes}, 0.664\colon \mathit{no}\}$$

The idea of those kinds of definitions is that they leave part of the probability mass out to express imprecision. The above definition is actually an abbreviation for:

$$\begin{aligned}\mathbf{DM\_if\_HighRisk} \sim \{\ & 0.266\colon \mathbf{DM\_if\_HighRisk} = \mathit{yes},\\ & 0.664\colon \mathbf{DM\_if\_HighRisk} = \mathit{no},\\ & 0.07\ \ \colon \mathbf{DM\_if\_HighRisk} \in \{\mathit{yes}, \mathit{no}\}\ \}\end{aligned}$$

Definitions can also be continuous distributions, such as $\mathcal{N}(0.0, 1.0)$. This can be seen as infinite definitions. In such a case, only approximations by finite definitions are computable, as will later be discussed in Section 4.5.2.5.

4.4.2 *Constraint Theory Examples*

PCLP can be based on arbitrary constraint theories under mild restrictions. Those restrictions will later be discussed in detail in Section 4.5.2.4, but a large class of practically useful constraints, such as inequalities on integers or real numbers, fulfil that requirement. We refer to an instance of PCLP based on constraint language **Constr** as PCLP(**Constr**). In this chapter we make use of two constraint theories which were already used in the examples before: real numbers (*R*) and discrete constants (*D*). In the examples of Section 4.3 and further in this chapter we use a combination and refer to the language as PCLP(*D*, *R*). The combination of different theories is realised by assuming that a constraint theory is attached to each variable. Statements about variables can only be made using the constraint language of the corresponding constraint theory attached. For example, real-valued random variables cannot be compared to discrete ones. Constraints of different theories can only be combined using logical connectives.

In the constraint theory *D* random variables take values of discrete constants. The basic building blocks of the language are set membership ($\in$), its negation ($\notin$) and their special cases equality ($=$) and inequality ($\neq$). The constraint theory is very similar to the *finite domain* constraint theory of CLP(FD) [66], with the difference that we do not require to explicitly define the range. The range of variables are all possible, countably-infinitely many constants. One can however effectively restrict the range by assigning probability mass to only a finite number of constants. Using *D*, one can represent, for instance, *Bayesian networks* (BNs) [119], but also first-order formalisms such as *causal probabilistic logic* (CP-logic) [152].

The R theory is basically the same as in CLP(R) [73]. Variables represent real numbers and constraints consist of linear equalities and inequalities using predicates such as $=, \neq, <, >, \leq, \geq$, rational numbers as constants and functions $+, -, \cdot$. One argument of the multiplication operator must be a constant to make constraints linear and guarantee decidability. This theory can be used to approximate a large class of continuous distributions, as will be shown in Section 4.5.2.5.

4.5 SEMANTICS

We develop the semantics of PCLP based on a more general, abstract semantics. For this we first generalise the distribution semantics and develop general conditions under which exact inference is possible (Section 4.5.1). We then extend this semantics with the idea of credal sets ensuring all discussed, novel properties of PCLP (Section 4.5.2). Those properties are proven for the abstract semantics. Based on those general results, we define the concrete semantics of PCLP and show its properties (Section 4.5.3).

4.5.1 *A Generalised Distribution Semantics*

In this section we introduce a generalised distribution semantics, extending Sato's original distribution semantics to random variables with arbitrary, possibly infinite ranges. While the original distribution semantics is defined for binary *probabilistic facts* only, it can easily be generalised to random variables with finite ranges, e.g. the implementation of the *PRISM* language supports such random variables [139]. However, as soon as we deal with infinite ranges, the generalisation is semantically far from straightforward, in particular when the ranges are uncountable. For example, grounding a real variable would lead to an uncountable number of ground atoms, for which the original distribution semantics does not provide a well-defined probability distribution.

We tackle this problem by augmenting our logical formalism with special constraints, an approach also adopted by deterministic CLP [72] and *satisfiability modulo theories* (SMT) solvers. This leads to an expressive language, where for many queries there is no closed form expression to compute *marginal* probabilities, i.e. exact inference is not possible. In the second part of this section, we therefore also discuss sufficient conditions under which exact inference is possible in this extended language.

Note that the results of this section are not fundamentally different from known solutions as offered by, for example, *Hybrid ProbLog* [60]. However, for the first time we formalise such extension in a general way. As will become clear, this more general theory will act as a basis for the more advanced work discussed in the remainder of the thesis.

4.5.1.1 *Probability Distributions on Constraint Logic Programs*

Whereas Sato's distribution semantics assigns a joint probability distribution to the ground atoms of a logic program using probabilistic facts, in the generalised distribution semantics we make use of constraints to define this joint distribution. Therefore, we make use of rules, which may include constraints, as introduced in Section 4.4.1. We introduce a generalised distribution semantics for this constraint logical language.

CONSTRAINT LOGIC THEORIES WITH PROBABILITY MEASURES The basic idea of the language is to have countably many random variables

$$\mathbf{V} = \{\mathbf{V}_1, \mathbf{V}_2, \ldots\},$$

with ranges that are sets of elements with arbitrary properties, for example discrete constants and real numbers. Hence, the number of random variables can be infinite, similar to the original distribution semantics. Note that probabilities of events involving an infinite number of variables may not be computable, but we separate the semantics from the issue of performing inference in order to not unnecessarily restrict the generality of the semantics.

Constraints φ will be looked upon as predicates on the state of the random variables, i.e. φ is a function from $\mathbf{V}_1 = v_1, \mathbf{V}_2 = v_2, \ldots$ to $\{\mathit{true}, \mathit{false}\}$ where v_i is an element in the range of $\mathbf{V}_i$. Equivalently, each constraint can be seen as a predicate on the *sample space*, as it corresponds to the random variables' states. The subset of the sample space where a constraint holds, is called the *solution space* of the constraint.

Definition 4.1 (Constraint Solution Space). *The solution space of a constraint φ given sample space Ω is defined as:*

$$\mathrm{CSS}(\varphi) \overset{\mathrm{def}}{=} \{\omega \in \Omega \mid \mathbf{V}(\omega) = \mathbf{v}, \varphi(\mathbf{v})\}$$

where $\mathbf{V}(\omega) = \mathbf{v}$ is shorthand for $\mathbf{V}_i(\omega) = \mathbf{v}_i$ for all i.

Formally, we define constraint logic theories as follows.

Definition 4.2 (Probabilistic Constraint Logic Theory). *A probabilistic constraint logic theory $\mathbf{T}$ is a tuple*

$$(\mathbf{V}, \Omega_{\mathbf{V}}, \mathcal{A}_{\mathbf{V}}, P_{\mathbf{V}}, \mathbf{Constr}, \mathbf{L}),$$

where

- $\mathbf{V} = \{\mathbf{V}_1, \mathbf{V}_2, \ldots\}$ *is a countable set representing random variables with associated non-empty ranges $\{\mathbf{Range}_1, \mathbf{Range}_2, \ldots\}$ (we fix the enumeration such that each random variable has an index);*
- $\Omega_{\mathbf{V}}$ *is the sample space of the random variables $\mathbf{V}$ defined as the Cartesian product of the random variables' ranges:*

$$\Omega_{\mathbf{V}} \overset{\mathrm{def}}{=} \mathbf{Range}_1 \times \mathbf{Range}_2 \times \cdots ;$$

- $\mathcal{A}_{\mathbf{V}}$ *is an* event space *representing events concerning the random variables* $\mathbf{V}$*;*
- $P_{\mathbf{V}}$ *is a* probability measure *on the space defined above, thus* $(\Omega_{\mathbf{V}}, \mathcal{A}_{\mathbf{V}}, P_{\mathbf{V}})$ *forms a* probability space*;*
- **Constr** *is a set of constraints, closed under conjunction, disjunction and negation, such that:*

  $$\{\text{CSS}(\varphi) \mid \varphi \in \mathbf{Constr}\} \subseteq \mathcal{A}_{\mathbf{V}},$$

  i.e. the constraints correspond to events;
- **L** *is a set of LP rules with constraints, as introduced in Section 4.4.1:*

  $$h \leftarrow l_1, \dots, l_n, \langle \varphi_1(\mathbf{V}) \rangle, \dots, \langle \varphi_m(\mathbf{V}) \rangle,$$

  where $\varphi_i \in \mathbf{Constr}, 1 \leq i \leq m$*.*

Note that since the sample space is defined by the Cartesian product of the ranges, the random variables are simple projections of single tuple elements of the sample space. So the solution space of an arbitrary constraint φ equals $\{\omega \in \Omega \mid \varphi(\omega)\}$.

In the remainder of this chapter, we abstract from the actual constraint language used. In the examples, the language used to define constraints is only meant as illustration. For example, if all $\mathbf{V}_i$ are continuous variables, then we may write $\langle \forall_i \; \mathbf{V}_{i+1} \geq \mathbf{V}_i \rangle$ to express that $\mathbf{V}_i$ increases with i.

Example 4.6

Consider the running example of Section 4.3.3:

$$\begin{aligned}
&\mathbf{Time\_Comp}_1 \sim \text{Exp}(1)\\
&\mathbf{Time\_Comp}_2 \sim \text{Exp}(1)\\
&\mathit{saved} \leftarrow \langle \mathbf{Time\_Comp}_1 < 0.75 \rangle\\
&\mathit{saved} \leftarrow \langle \mathbf{Time\_Comp}_1 < 1.25 \rangle, \langle \mathbf{Time\_Comp}_1 + 0.25 \cdot \mathbf{Time\_Comp}_2 < 1.375 \rangle
\end{aligned}$$

Suppose that $\mathbf{Time\_Comp}_1$ and $\mathbf{Time\_Comp}_2$ are the only two random variables in the enumeration. The range of both variables are the real numbers: $\mathbf{Range}_1 = \mathbf{Range}_2 = \mathbb{R}$. The used constraint language includes at least linear inequalities and the probability measure is such that the first two variables are independently distributed according to an exponential distribution.

EXTENDING PROBABILITY SPACES TO THE ENTIRE THEORY While a theory **T** defines a probability distribution over the random variables, the probabilities of the events in the logical language are not specified directly. As in the original distribution semantics, a probability distribution over the random variables can uniquely be extended to a distribution over the entire program. We show how to extend the sample space, the event space and the probability measure, such that it includes the atoms defined by the rules **L** as well.

The basic idea our generalisation is based on, is to look upon probabilistic facts, occurring in rules in the original distribution semantics, as a special kind of constraints, i.e. the probabilistic fact *pf* in a rule represents the constraint that *pf* is true. In order to generalise this semantics for arbitrary constraints, we extend the sample space by considering all atoms appearing in the logical theory **L**. We assume that there is a countable number of atoms, treat them as random variables taking values *true* or *false* and denote the set of all those atoms with **A**. As for variables in **V** we fix the enumeration and define the sample space $\Omega_{\mathbf{L}}$ being the Cartesian product of the values of all atoms:

$$\Omega_{\mathbf{L}} \overset{\text{def}}{=} \prod_{i=1}^{|\mathbf{A}|} \{\mathit{true}, \mathit{false}\} \tag{4.1}$$

For the event space of the logic part $\mathcal{A}_{\mathbf{L}}$, we can take the sample space's powerset, since the sample space is countable:

$$\mathcal{A}_{\mathbf{L}} \overset{\text{def}}{=} \wp(\Omega_{\mathbf{L}}) \tag{4.2}$$

The sample space for the entire theory $\Omega_{\mathbf{T}}$ is the Cartesian product of the sample spaces for the random variables and the logical theory:

$$\Omega_{\mathbf{T}} \overset{\text{def}}{=} \Omega_{\mathbf{V}} \times \Omega_{\mathbf{L}} \tag{4.3}$$

The event space of the entire theory $\mathcal{A}_{\mathbf{T}}$ is built as well from the event spaces of the random variables and logical theory. Concretely, it is their *tensor-product σ-algebra*:

$$\mathcal{A}_{\mathbf{T}} \overset{\text{def}}{=} \mathcal{A}_{\mathbf{V}} \otimes \mathcal{A}_{\mathbf{L}} \tag{4.4}$$

The tensor-product σ-algebra $\mathcal{A}_{\mathbf{V}} \otimes \mathcal{A}_{\mathbf{L}}$ is the smallest *σ-algebra* generated by the products of elements of $\mathcal{A}_{\mathbf{V}}$ and $\mathcal{A}_{\mathbf{L}}$: $\sigma_{\Omega_{\mathbf{V}} \times \Omega_{\mathbf{L}}}\big(\{e_{\mathbf{V}} \times e_{\mathbf{V}} \mid e_{\mathbf{V}} \in \mathcal{A}_{\mathbf{L}}, e_{\mathbf{L}} \in \mathcal{A}_{\mathbf{L}}\}\big)$. We cannot simply use the product of both spaces, as such product is not necessarily a σ-algebra.

Example 4.7

Consider the following event spaces:

$$\mathcal{A}_{\mathbf{V}} = \{\emptyset, \{a\}, \{b, c\}, \{a, b, c\}\}$$
$$\mathcal{A}_{\mathbf{L}} = \{\emptyset, \{\mathit{true}\}, \{\mathit{false}\}, \{\mathit{true}, \mathit{false}\}\}$$

The product $\{e_{\mathbf{V}} \times e_{\mathbf{L}} \mid e_{\mathbf{V}} \in \mathcal{A}_{\mathbf{V}}, e_{\mathbf{L}} \in \mathcal{A}_{\mathbf{L}}\}$ is then:

$$\begin{aligned}\{\ &\emptyset, \{(a, \mathit{true})\}, \{(a, \mathit{false})\}, \{(a, \mathit{true}), (a, \mathit{false})\},\\ &\{(b, \mathit{true}), (c, \mathit{true})\}, \{(b, \mathit{false}), (c, \mathit{false})\},\\ &\{(b, \mathit{true}), (b, \mathit{false}), (c, \mathit{true}), (c, \mathit{false})\},\\ &\{(a, \mathit{true}), (b, \mathit{true}), (c, \mathit{true})\}, \{(a, \mathit{false}), (b, \mathit{false}), (c, \mathit{false})\},\\ &\{(a, \mathit{true}), (a, \mathit{false}), (b, \mathit{true}), (b, \mathit{false}), (c, \mathit{true}), (c, \mathit{false})\}\ \}\end{aligned}$$

This product is no σ-algebra as for instance $\{(a, \mathit{false})\} \cup \{(b, \mathit{true}), (c, \mathit{true})\}$ is not included. In the tensor product the minimal number of elements are added to the product, such that it becomes a σ-algebra.

Finally, the probability measure $P_{\mathbf{V}}$ is extended to a probability measure on the entire theory yielding $P_{\mathbf{T}}$. The way this is achieved in the distribution semantics builds upon the observation that determining truth values of the probabilistic facts uniquely determines the truth values of all atoms in the entire theory. In our generalised setting, we similarly observe that determining which constraints hold uniquely determines the truth values of all atoms in the entire theory.

To formalise this notion, we will make use of the set $\mathbf{satisfiable}(\omega_{\mathbf{V}})$, which includes all constraints which are satisfiable given a *valuation* $\omega_{\mathbf{V}}$ of the random variables. We can interpret this set as a partial logic program, by assuming that each constraint occurs as instantiation of the predicate $\langle\rangle$.

Example 4.8

Suppose that $\omega_{\mathbf{V}} = (0, \ldots)$, i.e. the first random variable $\mathbf{V}_1$ takes value 0. Then $\mathbf{satisfiable}(\omega_{\mathbf{V}})$ does not include $\langle \mathbf{V}_1 > 0 \rangle$, but does include for instance $\langle \mathbf{V}_1 > -1 \rangle$.

An element of the sample space $\omega_{\mathbf{V}}$ therefore yields a LP theory, which is the combination of the logical rules provided by the theory $\mathbf{T}$ and all satisfiable constraints: $\mathbf{L} \cup \mathbf{satisfiable}(\omega_{\mathbf{V}})$. As discussed in Section 2.1.3, there are different ways to assign models to logic programs, but the goal is usually to have a single unique model. We abstract from which class of programs and which declarative semantics of LP are used; it is only required that each program has a unique model. Thus, in the following we use $M_{\mathbf{L}}(\omega_{\mathbf{V}})$ to denote the model given the theory, value assignments to random variables and the chosen semantics, although not necessarily the *least Herbrand model*.

Example 4.9

Consider again the rules of the running example (Section 4.3.3):

$$\mathit{saved} \leftarrow \langle \mathbf{Time\_Comp}_1 < 0.75 \rangle$$
$$\mathit{saved} \leftarrow \langle \mathbf{Time\_Comp}_1 < 1.25 \rangle, \langle \mathbf{Time\_Comp}_1 + 0.25 \cdot \mathbf{Time\_Comp}_2 < 1.375 \rangle$$

The definition of the atom *saved* implies that it is true if and only if the constraint $\mathbf{Time\_Comp}_1 < 0.75$ or both $\mathbf{Time\_Comp}_1 < 1.25$ and $\mathbf{Time\_Comp}_1 + 0.25 \cdot \mathbf{Time\_Comp}_2 < 1.375$ are satisfied.

The model $M_{\mathbf{L}}\big((1,2)\big)$ does not include the constraints $\langle \mathbf{Time\_Comp}_1 < 0.75 \rangle$ and $\langle \mathbf{Time\_Comp}_1 + 0.25 \cdot \mathbf{Time\_Comp}_2 < 1.375 \rangle$, since these constraints are not in $\mathbf{satisfiable}\big((1,2)\big)$. $\langle \mathbf{Time\_Comp}_1 < 1.25 \rangle$ is included in the model, but given the clauses above, *saved* is not included in $M_{\mathbf{L}}\big((1,2)\big)$. In contrast, the model $M_{\mathbf{L}}\big((0.5,1)\big)$ includes $\langle \mathbf{Time\_Comp}_1 < 0.75 \rangle$ and therefore *saved* as well.

Given these notions, a probability measure $P_{\mathbf{T}}$ can be defined on the extended event space. The idea is to uniquely derive this measure from $P_{\mathbf{V}}$ by mapping elements of the entire event space to elements of the random variables' event space, such that truth values of logical atoms correspond to the unique models $M_{\mathbf{L}}(\omega_{\mathbf{V}})$, given by valuations of random variables $\omega_{\mathbf{V}}$.

Definition 4.3 (Entire Theory Probability Measure)**.** *The probability measure on the entire theory* $\mathbf{T}$*'s event space is defined as:*

$$P_{\mathbf{T}}(e) \overset{\text{def}}{=} P_{\mathbf{V}}\big(\{\omega_{\mathbf{V}} \mid (\omega_{\mathbf{V}}, \omega_{\mathbf{L}}) \in e,\ M_{\mathbf{L}}(\omega_{\mathbf{V}}) \models \omega_{\mathbf{L}}\}\big)$$

Here we use elements $\omega_{\mathbf{L}}$ of the logical theory's sample space as logical formulas, where they represent conjunctions determining for each atom whether it is true or not. For example, $\omega_{\mathbf{L}} = (0, 1, 0, \ldots)$ means $\neg a \land b \land \neg c \land \cdots$, where $a, b, c, \ldots$ are the atoms of $\mathbf{L}$.

It is ensured that events in this definition are in $\mathcal{A}_{\mathbf{V}}$, because the restriction $M_{\mathbf{L}}(\omega_{\mathbf{V}}) \models \omega_{\mathbf{L}}$ is based on compositions of events from $\mathcal{A}_{\mathbf{V}}$. We then extend this to the probability of a query atom q given a probability measure $P_{\mathbf{T}}$ as follows.

Definition 4.4 (Query Probability)**.** *The probability of query q is defined as:*

$$P(q) \overset{\text{def}}{=} P_{\mathbf{T}}\big(\{(\omega_{\mathbf{V}}, \omega_{\mathbf{L}}) \in \Omega_{\mathbf{T}} \mid \omega_{\mathbf{L}} \models q\}\big)$$

Note that there is no need for a restriction on the values of random variables in $\omega_{\mathbf{V}}$ in Definition 4.4, since Definition 4.3 ensures that valuations of random variables for which q does not hold do not contribute to the probability. We further know that the event defined by the equation above is an element of the event space $\mathcal{A}_{\mathbf{T}}$, since we do not put any restrictions on values of random variables and the event space concerning the logic atoms is defined as the powerset of the sample space (Equation 4.2) thus each subset of the sample space is in the event space.

Example 4.10

Consider again the rules of the running example (Section 4.3.3):

$$saved \leftarrow \langle \mathbf{Time\_Comp_1} < 0.75 \rangle$$
$$saved \leftarrow \langle \mathbf{Time\_Comp_1} < 1.25 \rangle, \langle \mathbf{Time\_Comp_1} + 0.25 \cdot \mathbf{Time\_Comp_2} < 1.375 \rangle$$

Suppose that *saved* corresponds to the first dimension in $\Omega_{\mathbf{L}}$, such that $\omega_{\mathbf{L}} \models$ *saved* requires the first element of each sample to be *true*. Then by applying Definition 4.4 we obtain:

$$P(saved) = P_{\mathbf{T}}\big(\{(\omega_{\mathbf{V}}, (saved, \ldots)) \in \Omega_{\mathbf{T}} \mid saved = true\}\big)$$

Then by applying Definition 4.3 we see that:

$$\begin{aligned} P(saved) = P_{\mathbf{V}}\big(\{(\omega_1, \ldots) \mid ((\omega_1, \ldots), (saved, \ldots)) \in \Omega_{\mathbf{T}},& \\ M_{\mathbf{L}}(\omega_1, \ldots) \models (saved, \ldots),\ saved = true\}\big)& \end{aligned}$$

Given the clauses above and assuming **Time_Comp**$_1$ and **Time_Comp**$_2$ correspond to the first two random variables, $M_{\mathbf{L}}(\omega_1, \omega_2, \dots)$ entails *saved* if and only if $\omega_1 < 0.75 \lor (\omega_1 < 1.25 \land \omega_1 + 0.25 \cdot \omega_2 < 1.375)$:

$$\begin{aligned}
&P(\mathit{saved})\\
&\quad = P_{\mathbf{V}}\big(\{(\omega_1, \dots) \mid ((\omega_1, \dots), (\mathit{saved}, \dots)) \in \Omega_{\mathbf{T}},\\
&\qquad\qquad\qquad \omega_1 < 0.75 \lor (\omega_1 < 1.25 \land \omega_1 + 0.25 \cdot \omega_2 < 1.375)\}\big)\\
&\quad = P_{\mathbf{V}}\big(\{(\omega_1, \dots) \in \Omega_{\mathbf{V}} \mid \omega_1 < 0.75 \lor (\omega_1 < 1.25 \land \omega_1 + 0.25 \cdot \omega_2 < 1.375)\}\big)
\end{aligned}$$

As **Time_Comp**$_1$, **Time_Comp**$_2$ $\sim \mathrm{Exp}(1)$, the probability $P(\mathit{saved})$ can be computed as follows. We denote the associated PDF with f and the *cumulative distribution function* (CDF) with F:

$$\begin{aligned}
&P_{\mathbf{V}}(\{(\omega_1, \dots) \in \Omega_{\mathbf{V}} \mid \omega_1 < 0.75\}) = F(0.75) = 1 - e^{-0.75} \approx 0.53\\
&P_{\mathbf{V}}(\{(\omega_1, \dots) \in \Omega_{\mathbf{V}} \mid \omega_1 < 0.75 \land \omega_1 + 0.25 \cdot \omega_2 < 1.375\})\\
&\quad = \int_0^{0.75} f(x) P(\mathbf{Time\_Comp}_2 < 4 \cdot 1.375 - 4x) \mathrm{d}x\\
&\quad = \int_0^{0.75} f(x) F(5.5 - 4x) \mathrm{d}x = \int_0^{0.75} e^{-x} - e^{3x-5.5} \mathrm{d}x \approx 0.52\\
&P_{\mathbf{V}}(\{(\omega_1, \dots) \in \Omega_{\mathbf{V}} \mid \omega_1 < 1.25 \land \omega_1 + 0.25 \cdot \omega_2 < 1.375\})\\
&\quad = \int_0^{1.25} e^{-x} - e^{3x-5.5} \mathrm{d}x \approx 0.66
\end{aligned}$$

With this, we can now compute the probability of *saved* as:

$$\begin{aligned}
&P_{\mathbf{V}}(\{(\omega_1, \dots) \in \Omega_{\mathbf{V}} \mid \omega_1 < 0.75 \lor (\omega_1 < 1.25 \land \omega_1 + 0.25 \cdot \omega_2 < 1.375)\})\\
&\qquad \approx 0.53 + 0.66 - 0.52 = 0.67
\end{aligned}$$

By combining Definitions 4.3 and 4.4 we can determine a single event in the random variables' probability space with the same probability as a query q. Such an event is referred to in the following as the *solution event*. In this way, a separation is achieved between the symbolic part – determining in which cases a statement is true by logical reasoning – and the probabilistic part – determining the probability that one of those cases occurs.

Definition 4.5 (Solution Event)**.** *The solution event of a query q is defined as:*

$$\mathrm{SE}(q) \overset{\mathrm{def}}{=} \{\omega_{\mathbf{V}} \in \Omega_{\mathbf{V}} \mid M_{\mathbf{L}}(\omega_{\mathbf{V}}) \models q\}$$

Lemma 4.1. *The probability of a query q as defined by Definition 4.4 can be computed using the solution event:*

$$P(q) = P_{\mathbf{V}}\big(\mathrm{SE}(q)\big)$$

All proofs are provided in Appendix A.

Definition 4.5 formulates the solution event in terms of samples. A different formulation in terms of constraints is, however, more suited for the subsequent formulation of conditions under which exact inference is feasible. Such a formulation is provided by the following lemma.

Lemma 4.2. *The solution event of a query q can be expressed as:*

$$\mathrm{SE}(q) = \mathrm{CSS}\Big(\bigvee_{\substack{M_{\mathbf{L}}[\Phi] \models q \\ \Phi \subseteq \mathbf{Constr}}} \bigwedge_{\varphi \in \Phi} \varphi \Big),$$

where Φ denotes subsets of the set of all constraints and $M_{\mathbf{L}}[\Phi]$ the model of the theory $\mathbf{L} \cup \{\langle\varphi\rangle \mid \varphi \in \Phi\}$.

Intuitively, the disjunction represents the collection of all possible models in which q is true in a logical sense, without dealing with the meaning of the construct $\langle\rangle$. All constraints in each Φ must be true at the same time in order to make q true as well. Thus using conjunctions, the sets of constraints are combined.

Example 4.11

Consider again the clauses of the running example (Section 4.3.3):

$$saved \leftarrow \langle \mathbf{Time\_Comp_1} < 0.75 \rangle$$
$$saved \leftarrow \langle \mathbf{Time\_Comp_1} < 1.25 \rangle, \langle \mathbf{Time\_Comp_1} + 0.25 \cdot \mathbf{Time\_Comp_2} < 1.375 \rangle$$

Given these two clauses, there are two ways to prove *saved*. Each model including *saved* therefore has to include all constraints enforced by the first or second clause, which corresponds to the disjunction in Lemma 4.2. Actually, in the definition above also all subsets of both cases are considered, which are however redundant, given that they are combined as disjunction. The second clause requires two constraints to hold, so they are combined using the conjunction in Lemma 4.2. The solution event of *saved* therefore is $\{(\omega_1, \omega_2, \ldots) \in \Omega_{\mathbf{V}} \mid \omega_1 < 0.75 \vee (\omega_1 < 1.25 \wedge \omega_1 + 0.25 \cdot \omega_2 < 1.375)\}$. Note that the event is the same as used to compute the query probability in Example 4.10.

4.5.1.2 *Exact Inference Conditions*

The semantics introduced above is very general and powerful, but exact computation of event probabilities is in general not possible. Practically useful languages always demand finding the proper balance between expressivity and feasibility of inference. We therefore discuss ways to restrict the general semantics in such a way that we can perform exact inference. The result provides a basis for analysing in a structured way, which properties allow exact inference for different languages.

Proposition 4.1. *The probability of an arbitrary query can be computed exactly if the following conditions hold:*

1. **finite-relevant-constraints condition**: *There are only finitely many constraint predicates ($\langle\rangle$) relevant for determining truthfulness of each query atom. Formally, a constraint predicate $\langle\varphi\rangle$ is relevant for a query atom q if there exists a set of constraint predicates Φ, such that $q \in M(\Phi \cup \mathbf{L}) \not\Leftrightarrow q \in M(\{\langle\varphi\rangle\} \cup \Phi \cup \mathbf{L})$. Intuitively, there exists a set of constraint predicates for which it matters whether $\langle\varphi\rangle$ is included in the program or not, meaning $\langle\varphi\rangle$ is relevant. We also assume that finding such relevant constraints predicates and entailment checking can be done in finite time.*

2. **finite-dimensional-constraints condition**: *Constraints occurring in clauses as argument of the construct $\langle\rangle$ only concern a finitely-dimensional subset of the sample space. This means the constraints' solution spaces have the form:*

   $$\{(\omega_1, \dots, \omega_n, \omega_{n+1}, \dots) \in \Omega_{\mathbf{V}} \mid cond(\omega_1, \dots, \omega_n)\},$$

   where cond is an arbitrary predicate with n arguments, i.e. the constraint puts a condition only on a finite number of variables.

3. **computable-measure condition**: *The probabilities of finite-dimensional events, in the sense of the previous condition, are computable.*

The computable-measure condition implies that one can exactly compute finite-dimensional integrals over employed PDFs, which is only possible under very strong assumptions.

The conditions stated here are sufficient, but not strictly necessary, as we do not restrict the kind of continuous distributions employed, for instance to Gaussian distributions, which is often done in other work, as discussed in Section 4.6. These conditions generalise the restrictions typically enforced by other approaches based on the distribution semantics, as will be discussed below. The following example illustrates the exact inference conditions.

Example 4.12

Consider again the clauses of the running example (Section 4.3.3):

$$saved \leftarrow \langle \mathbf{Time\_Comp}_1 < 0.75\rangle$$
$$saved \leftarrow \langle \mathbf{Time\_Comp}_1 < 1.25\rangle, \langle \mathbf{Time\_Comp}_1 + 0.25 \cdot \mathbf{Time\_Comp}_2 < 1.375\rangle$$

As shown in Example 4.11 the solution event can be derived in finite time (finite-relevant-constraints condition) and is finite-dimensional, since all constraints in the program above are finite-dimensional as well (finite-dimensional-constraints condition). We can, as shown in Example 4.10, exactly compute that the probability of $\{(\omega_1, \omega_2, \dots) \in \Omega_{\mathbf{V}} \mid \omega_1 < 0.75 \vee (\omega_1 < 1.25 \wedge \omega_1 + 0.25 \cdot \omega_2 < 1.375)\}$ is 0.67 (computable-measure condition), thus $P(saved) = 0.67$. This is only possible, because we use exponential distributions. Employing other, possible different distributions, for the random variables, would however quickly break the computable-measure condition.

An example of a program that breaks the exact inference condition is:

$$forever\_sun(X) \leftarrow \langle \mathbf{Weather}_X = sunny\rangle, forever\_sun(X+1)$$

The predicate *forever_sun*(X) intuitively represents that it is sunny forever from day X on, assuming there are infinitely many days in the future. The probability of *forever_sun*(0) cannot be computed in finite time, since an infinite number of days have to be considered, which means the finite-relevant-constraints condition is violated. One could usually say the limit of the probability of *forever_sun*(0) would be 0, but this is not true for all possible probability measures. Only assuming the computable-measure condition for the probability measure one cannot draw that conclusion.

An alternative definition of the problem would be:

$$\mathit{forever\_sun} \leftarrow \langle \forall_{i \in \mathbb{N}}\ \mathbf{Weather}_i = \mathit{sunny} \rangle$$

Similarly, the probability of *forever_sun* might be computable with further assumptions on the probability space, but generally this is not possible, since the finite-dimensional-constraints condition is violated.

We briefly discuss those conditions in the context of existing languages. The finite-relevant-constraints condition seems a very reasonable condition for allowing exact inference and cannot be avoided. In Sato's original semantics, a condition called *finite-support condition* is required for probabilistic facts, which is similar to the finite-relevant-constraints condition, although restricted to positive programs. Similarly, the finite-dimensional-constraints condition is enforced in Sato's semantics, if we interpret probabilistic facts in the program as constraints concerning only a single variable, i.e. a probabilistic fact is required to be true or false; dependencies are expressed by the structure of rules. So actually a stronger variant of the finite-dimensional-constraints condition is enforced, as constraints only concern single variables. Finally, the computable-measure condition depends on how the probability distribution is defined in a concrete language. Most languages based on the distribution semantics satisfy this property by assuming that all random variables are (mutually) independent. This means that the probability measure is defined in terms of a single probability per variable and consequently the probability of events consisting of only a finite number of variables can be computed in finite time. Again, this is done without loss of generality, as dependencies can be introduced by the structure of the logic program. The situation is more complex as soon as continuous distributions are considered. A language allowing for continuous variables and nevertheless exact inference is Hybrid ProbLog [60] which extends Sato's semantics for continuous variables. As in Sato's semantics, Hybrid ProbLog only allows constraints on single variables. This means that for instance $\langle \mathbf{V}_1 > 0 \rangle$ is allowed, but $\langle \mathbf{V}_1 \geq \mathbf{V}_2 \rangle$ is not. This restriction ensures that the computable-measure condition is fulfilled, under the assumption that we can compute CDFs of the employed continuous distributions. While in the binary case the restriction to constraints on single variables does not restrict expressiveness of the language, in the continuous case it does. A way to overcome these restrictions is discussed next.

4.5.2 *Defining Credal Sets*

The aim of this section is to provide a new theory that supports exact inference also covering problems involving constraints with multiple variables. As a consequence, only the finite-dimensional-constraints condition is required, as introduced in the previous section. For example, we wish to allow the comparison of real-valued random variables, while at the same time avoiding severely restricting the kind of distributions to fulfil the computable-measure condition. This problem is tackled by introducing credal sets, i.e. a set of probability distributions. We show that this idea makes it possible to compute bounds on the probabilities of a query under conditions that are less strict than those assumed before. We finally discuss an important application of the theory: the approximation of precise, continuous distributions.

4.5.2.1 *Credal Set Specifications*

As discussed before, the basic idea is to assign probability masses to sets of values. Sets of values correspond to events and consequently credal sets are defined in terms of probability-event pairs. We refer to this kind of definitions as *credal set specifications*. Such specifications have the desirable property that they are guaranteed to define non-empty credal sets, i.e. sets of probability measures. Credal set specifications introduced in this section are between the purely semantic level of credal sets and the concrete syntactic level, which is discussed later.

Since the number of random variables in our semantics can be infinite, we would have to allow potentially infinite sets of probability-event pairs. Such specifications would be hard to define directly and it is not clear how to construct probability distributions consistent with such specifications. Therefore, we define credal set specifications by means of a sequence of countably, potentially infinite number of specifications, each defining finite-dimensional credal sets with increasing dimensionality. We have to make sure such specifications do not contradict each other, which we ensure by a property called *compatibility*. This allows us to use an existing construction theorem [110, III.3] to construct infinite-dimensional probability distributions, consistent with a credal set specification given.

Before we formally define the concept of credal set specifications we introduce the concept of *event projections*. We denote the i-th event projection of event e, where i is a natural number, with $\pi_i(e)$ and define it as:

$$\pi_i(e) \overset{\text{def}}{=} \{(\omega_1, \ldots, \omega_i) \mid (\omega_1, \ldots, \omega_i, \ldots) \in e\} \tag{4.5}$$

Event projections are also similarly defined for finite events.

Definition 4.6 (Credal Set Specification). *A credal set specification* $\mathbf{C}$ *consists of a countable number of finite-dimensional specifications* $\mathbf{C}_1, \mathbf{C}_2, \ldots$*. Each* $\mathbf{C}_k$ *has the form of a finite collection of probability-event pairs* $(p_1, e_1), (p_2, e_2), \ldots, (p_n, e_n)$ *such that for each* $\mathbf{C}_k$*:*

1. *The events are in a finite-dimensional event space* $\mathcal{A}^k_{\mathbf{V}}$ *over the sample space* $\Omega^k_{\mathbf{V}} \overset{\text{def}}{=} \mathbf{Range}_1 \times \mathbf{Range}_2 \times \cdots \times \mathbf{Range}_k$.
2. *The sum of the probabilities is* 1.0*:* $\sum\limits_{(p,e)\in \mathbf{C}_k} p = 1.0$.
3. *The events must not be the empty set:* $\forall_{(p,e)\in\mathbf{C}_k}\, e \neq \emptyset$.

Additionally, all finite $\mathbf{C}_k$ *must be compatible, i.e. for all k:* $\mathbf{C}_k = \pi_k(\mathbf{C}_{k+1})$*, where* $\pi_l(\mathbf{C}_k)$ *is defined as:*

$$\pi_l(\mathbf{C}_k) \overset{\text{def}}{=} \{(\sum_{\substack{\pi_l(e)=e' \\ (p,e)\in\mathbf{C}_k}} p, e') \mid e' \in \{\pi_l(e) \mid (p,e) \in \mathbf{C}_k\}\}$$

One can look upon these credal set specifications as a way to split the probability mass into portions which are assigned to specific non-empty events given a finite set of random variables. As said, compatibility is used to inductively construct consistent distributions over all random variables by ensuring that specifications of different dimensionality do not contradict each other.

Example 4.13

Assume the first two random variables in the sample space have as range the set $\{sun, rain\}$ and $\mathbf{C}_1 = \Big\{\big(0.2, \{sun, rain\}\big), \big(0.8, \{sun\}\big)\Big\}$. Suppose this means that the probability that there is sun tomorrow is at least 80%.

Consider the following specification, concerning not only the weather of tomorrow, but as well the weather of the day after tomorrow:

$$\mathbf{C}_2 = \Big\{\big(0.2, \{(sun, sun), (sun, rain)\}\big), \big(0.8, \{(sun, sun)\}\big)\Big\}$$

Both specifications are not compatible, as $\mathbf{C}_2$ fixes the probability that there is sun tomorrow to 1.0, which conflicts with $\mathbf{C}_1$, i.e. $\pi_1(\mathbf{C}_2) = \Big\{\big(1.0, \{sun\}\big)\Big\} \neq \mathbf{C}_1$. In contrast, an example of a compatible specification is:

$$\mathbf{C}'_2 = \Big\{\big(0.2, \{(sun, sun), (sun, rain), (rain, sun)\}\big), \big(0.8, \{(sun, sun)\}\big)\Big\}$$

Each $\mathbf{C}_k$ defines a set of probability measures, given by the following definition.

Definition 4.7 (Finite Credal Sets)**.** *The set of all probability measures* $\mathbf{P}^k_{\mathbf{V}}$ *defined by* $\mathbf{C}_k$ *includes all probability measures of* $\mathcal{A}^k_{\mathbf{V}}$ *consistent with Kolmogorov's probability axioms, for which additionally the following condition holds. A probability measure* $P_{\mathbf{V}}$ *is in* $\mathbf{P}^k_{\mathbf{V}}$ *if for each event* $e \in \mathcal{A}^k_{\mathbf{V}}$*:*

$$\sum_{\substack{d\subseteq e \\ (p,d)\in\mathbf{C}_k}} p \leq P_{\mathbf{V}}(e) \leq \sum_{\substack{d\cap e\neq\emptyset \\ (p,d)\in\mathbf{C}_k}} p$$

The probabilities contributing to the lower bound are related to events which are subsets of e, and therefore certainly have to contribute to the probability of e as well. In contrast, the probabilities contributing to the upper bound relate to events which are not disjoint with e, and therefore can possibly contribute to the probability of e.

Example 4.14

For illustration, we give a two-dimensional specification with two variables with range $\{sun, rain\}$:

$$\mathbf{C}_2 = \Big\{ \big(0.2, \{(sun, sun), (sun, rain)\}\big), \big(0.8, \{(sun, sun), (sun, rain), (rain, sun)\}\big) \Big\}$$

Some distributions which are element of the resulting credal set are:

	(sun, sun)	$(sun, rain)$	$(rain, sun)$	$(rain, rain)$
$P^1_{\mathbf{V}}$	1.0	0.0	0.0	0.0
$P^2_{\mathbf{V}}$	0.0	1.0	0.0	0.0
$P^3_{\mathbf{V}}$	0.5	0.5	0.0	0.0
$P^4_{\mathbf{V}}$	0.2	0.3	0.5	0.0
$P^5_{\mathbf{V}}$	0.1	0.1	0.8	0.0
...	...	...	...	...

With a more strict specification we restrict the possible measures further:

$$\mathbf{C}_2 = \Big\{ \big(0.2, \{(sun, sun), (sun, rain)\}\big), \big(0.8, \{(sun, sun)\}\big) \Big\}$$

In this case for all $P_{\mathbf{V}}$ in the credal set: $P_{\mathbf{V}}\big((rain, sun)\big) = P_{\mathbf{V}}\big((rain, rain)\big) = 0.0$, $0.0 \leq P_{\mathbf{V}}\big((sun, rain)\big) \leq 0.2$ and $P_{\mathbf{V}}\big((sun, sun)\big) = 1 - P_{\mathbf{V}}\big((sun, rain)\big)$.

Next, we show that a credal set specification $\mathbf{C}$ defines a credal set for the entire, potentially infinite-dimensional, random variables' sample space and show that this set is convex and non-empty. To prove this fundamental property we show all $\mathbf{C}_k$ of $\mathbf{C}$ define a non-empty set of probability spaces over the entire event space $\mathcal{A}_{\mathbf{V}}$, which satisfy Kolmogorov's probability axioms.

Lemma 4.3. *For each credal set specification $\mathbf{C} = \{\mathbf{C}_1, \mathbf{C}_2, \ldots\}$ there exists a non-empty credal set of probability measures $\mathbf{P_V}$ on the entire space $(\Omega_{\mathbf{V}}, \mathcal{A}_{\mathbf{V}})$, such that all measures $P_{\mathbf{V}}$ in $\mathbf{P_V}$ agree with $(\mathbf{C}_1, \mathbf{C}_2, \ldots)$ in the sense that for all events e and natural numbers k:*

$$\sum_{\substack{d \subseteq e \\ (p,d) \in \mathbf{C}_k}} p \leq P_{\mathbf{V}}\big(\pi_k(e)\big) \leq \sum_{\substack{d \cap e \neq \emptyset \\ (p,d) \in \mathbf{C}_k}} p$$

We require the additional condition that the event space is chosen such that it is possible to construct an infinite dimensional probability measure from an infinite number of finite ones with increasing dimensionality.

The technical condition which enables the construction of probability measures poses no practical restriction. For instance, the condition is fulfilled in case the event space for the random variables is built from *Borel σ-algebras* of *Polish topological spaces* [110, III.3]. This includes virtually all possible event spaces relevant for practical applications, such as all subsets of spaces with discrete topology, for instance integers or other sets of countably many constants, and all closed and open intervals with rational bounds in the real numbers.

Moreover, the credal set on the random variables can be extended to a credal set on the entire theory, by extending each element of the credal set as shown in Section 4.5.1.1.

Theorem 4.1. *Each credal set specification* $\mathbf{C}$ *defines a non-empty credal set of probability measures* $\mathbf{P_T}$ *on the entire theory and a set of corresponding query probability distributions* $\mathbf{P}$.

4.5.2.2 *Probability Bounds*

We have shown that a credal set specification defines a potentially infinite set of probability spaces, which means that for a query one can compute a potentially infinite number of probabilities. Instead, we are typically interested in the lower and upper bounds on the probability of a query.

Definition 4.8 (Probability Bounds). *We define the lower and upper probability bounds of a query q as follows:*

$$\underline{P}(q) \overset{\text{def}}{=} \min_{P \in \mathbf{P}} P(q)$$
$$\overline{P}(q) \overset{\text{def}}{=} \max_{P \in \mathbf{P}} P(q)$$

Furthermore, we introduce formulas for computing those bounds for finite-dimensional queries.

Proposition 4.2. *The lower and upper probability bounds of a query q, fulfilling the finite-dimensional-constraints condition and putting constraints only on the first k dimensions, can be computed by:*

$$\underline{P}(q) = \sum_{\substack{e \subseteq \mathrm{SE}(q) \\ (p,e) \in \mathbf{C}_k}} p$$
$$\overline{P}(q) = \sum_{\substack{e \cap \mathrm{SE}(q) \neq \emptyset \\ (p,e) \in \mathbf{C}_k}} p$$

Example 4.15

Consider again the clauses of the running example (Section 4.3.3):

$$saved \leftarrow \langle \mathbf{Time\_Comp_1} < 0.75 \rangle$$
$$saved \leftarrow \langle \mathbf{Time\_Comp_1} < 1.25 \rangle, \langle \mathbf{Time\_Comp_1} + 0.25 \cdot \mathbf{Time\_Comp_2} < 1.375 \rangle$$

We give a finite credal set specification for the two variables occurring in the clauses. This is done in such a way that it roughly matches the exponential distributions used in the example: $\mathbf{Time\_Comp_1}, \mathbf{Time\_Comp_2} \sim \text{Exp}(1)$. How to generate finite credal sets exactly matching continuous distributions in general is discussed in Section 4.5.2.5.

$$\begin{aligned}\mathbf{C}_2 = \{ \ &\big(0.49, \{(\omega_1, \omega_2) \mid 0 \le \omega_1 \le 1, 0 \le \omega_2 \le 1\}\big),\\ &\big(0.14, \{(\omega_1, \omega_2) \mid 1 \le \omega_1 \le 2, 0 \le \omega_2 \le 1\}\big),\\ &\big(0.07, \{(\omega_1, \omega_2) \mid 2 \le \omega_1 \le 3, 0 \le \omega_2 \le 1\}\big),\\ &\big(0.14, \{(\omega_1, \omega_2) \mid 0 \le \omega_1 \le 1, 1 \le \omega_2 \le 2\}\big),\\ &\big(0.04, \{(\omega_1, \omega_2) \mid 1 \le \omega_1 \le 2, 1 \le \omega_2 \le 2\}\big),\\ &\big(0.02, \{(\omega_1, \omega_2) \mid 2 \le \omega_1 \le 3, 1 \le \omega_2 \le 2\}\big),\\ &\big(0.07, \{(\omega_1, \omega_2) \mid 0 \le \omega_1 \le 1, 2 \le \omega_2 \le 3\}\big),\\ &\big(0.02, \{(\omega_1, \omega_2) \mid 1 \le \omega_1 \le 2, 2 \le \omega_2 \le 3\}\big),\\ &\big(0.01, \{(\omega_1, \omega_2) \mid 2 \le \omega_1 \le 3, 2 \le \omega_2 \le 3\}\big) \ \}\end{aligned}$$

The solution event of *saved* is $\{(\omega_1, \omega_2, \ldots) \in \Omega_{\mathbf{V}} \mid \omega_1 < 0.75 \lor (\omega_1 < 1.25 \land \omega_1 + 0.25 \cdot \omega_2 < 1.375)\}$, as shown in Example 4.11. The sample space together with the events defined in the credal set specification is shown in Figure 4.4. The solution event is visualised as line, where everything above the line is inside the solution event.

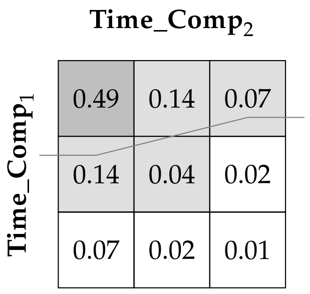


Figure 4.4: Two-Dimensional Sample Space with Credal Set Specification and Solution Event

The first event in $\mathbf{C}_2$ – shown in the left upper corner in Figure 4.4 – is a subset of the solution event. This intuitively means that no matter how we distribute the probability mass of 0.49 inside the event, all of it will be inside the solution event. Therefore $\underline{P}(\textit{saved}) = 0.49$. All events represented by grey areas in Figure 4.4 are not disjoint with the solution event. This means that it is possible to distribute the probability mass in such a way that all of it is inside the solution event. We can conclude $\overline{P}(\textit{saved}) = 0.49 + 0.14 + 0.07 + 0.14 + 0.04 = 0.88$.

We can finally provide formulas for the probability bounds for the general case that constraints are not finite-dimensional.

Theorem 4.2. *The lower and upper probability bounds of a query q can be expressed as:*

$$\underline{P}(q) = \lim_{k\to\infty} \sum_{\substack{e\subseteq \mathrm{SE}(q)\\(p,e)\in \mathbf{C}_k}} p$$
$$\overline{P}(q) = \lim_{k\to\infty} \sum_{\substack{e\cap \mathrm{SE}(q)\neq\emptyset\\(p,e)\in \mathbf{C}_k}} p$$

We finally present a result about the relation between lower and upper bound, which follows from this theorem and the sum rule for limits.

Corollary 4.1. *The lower bound can be expressed in terms of the upper bound and vice versa.*

$$\underline{P}(q) = 1 - \overline{P}(\neg q)$$
$$\overline{P}(q) = 1 - \underline{P}(\neg q)$$

4.5.2.3 *Dealing with Evidence*

Making use of evidence is crucial for probabilistic reasoning. Taking evidence into account means to exclude parts of the event space and renormalise the probability measure such that probabilities sum up to one. For a single probability distribution the probability of a query q given evidence e denoted by $P(q \mid e)$ can be expressed in terms of non-conditional probabilities as $P(q \wedge e)/P(e)$. In case of a credal set we have to do this for all corresponding conditional probabilities and find the minimum and maximum.

Definition 4.9 (Conditional Probability Bounds)**.** *We define the probability bounds given evidence as:*

$$\underline{P}(q \mid e) \overset{\text{def}}{=} \min_{P\in\mathbf{P}} P(q \mid e)$$
$$\overline{P}(q \mid e) \overset{\text{def}}{=} \max_{P\in\mathbf{P}} P(q \mid e)$$

Note that there are alternative definitions of conditional probabilities for imprecise probability distributions. Weichselberger argues that different notions of conditional probabilities should be used depending on the purpose they are used for [157]. We here restrict to our definition above, which corresponds to what Weichselberger calls the *intuitive concept*.

In contrast to the precise case we cannot define a normalisation factor – also called partition function – which only depends on the evidence, but also have to take the query into account. This is illustrated by the following example.

Example 4.16 ______

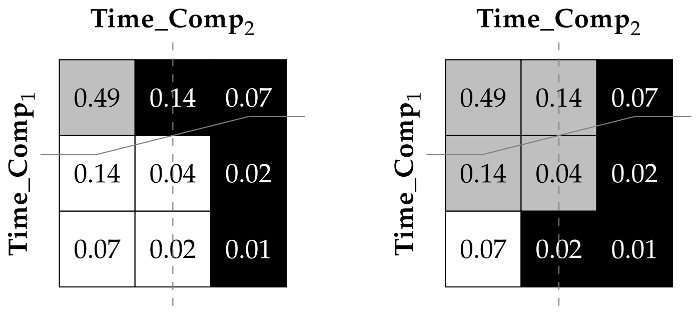
Consider again the clauses of the running example (Section 4.3.3):

$$\mathit{saved} \leftarrow \langle \mathbf{Time\_Comp_1} < 0.75\rangle$$
$$\mathit{saved} \leftarrow \langle \mathbf{Time\_Comp_1} < 1.25\rangle, \langle \mathbf{Time\_Comp_1} + 0.25 \cdot \mathbf{Time\_Comp_2} < 1.375\rangle$$

Consider an additional rule:

$$e \leftarrow \langle \mathbf{Time\_Comp_2} < 1.5 \rangle$$

Suppose we would like to compute $\overline{P}(saved \mid e)$. This is illustrated in Figure 4.5, where the events represented by black areas are excluded from the event space. All events in the right column certainly have to be excluded, since they are disjoint with the evidence. It depends on the probability measure chosen from the credal set whether to exclude or not the events in the middle column. The choice depends on whether we want to compute the lower or upper bound as is illustrated by the figure.



(a) Lower Bound (b) Upper Bound

Figure 4.5: Two-Dimensional Sample Space with Credal Set Specification, Solution Event (solid) and Evidence (dashed)

The probability mass of the upper middle event can only be included in the partition function (within e) if also completely within *saved*. So excluding it minimises the probability. The remaining events in the middle column would only contribute to the partition function of the lower bound. Excluding them would increase the result. The events are consequently not excluded for computing the lower bound, which is $\underline{P}(saved \mid e) = 0.49/(0.49 + 0.14 + 0.04 + 0.07 + 0.02) \approx 0.64$.

For the upper bound events contributing to the numerator and denominator of the probability have to be included, but events only contributing to the denominator have to be excluded, in order to obtain the maximal probability. The upper bound is thus $\overline{P}(saved \mid e) = (0.49 + 0.14 + 0.14 + 0.04)/(0.49 + 0.14 + 0.14 + 0.04 + 0.07) \approx 0.92$.

To compute conditional probabilities we have the following proposition that relates joint probabilities to conditional probabilities.

Proposition 4.3. *The probability bounds of a query q given evidence e, as defined by Definition 4.9, can be computed as follows:*

$$\underline{P}(q \mid e) = \frac{\underline{P}(q \wedge e)}{\underline{P}(q \wedge e) + \overline{P}(\neg q \wedge e)}$$

$$\overline{P}(q \mid e) = \frac{\overline{P}(q \wedge e)}{\overline{P}(q \wedge e) + \underline{P}(\neg q \wedge e)}$$

4.5.2.4 *Exact Inference Conditions*

We introduce a variant of the exact inference conditions (Proposition 4.1) concerning probability bounds instead of precise probabilities.

Theorem 4.3. *The probability bounds of an arbitrary query can be computed in finite time if the following conditions hold:*

1. ***finite-relevant-constraints condition**: There are only finitely many constraint predicates ($\langle\rangle$) relevant for determining truthfulness of each query atom and finding such relevant constraint predicates and entailment checking can be done in finite time. The condition is the same in Proposition 4.1.*

2. ***finite-dimensional-constraints condition**: Events occurring in clauses as argument of $\langle\rangle$ only concern a finitely-dimensional subset of the sample space. The condition is the same in Proposition 4.1.*

3. **disjoint-events-decidability condition**: *For each two finite-dimensional events e_1 and e_2 in the event space $\mathcal{A}_{\mathbf{V}}$ one can decide whether they are disjoint or not ($e_1 \cap e_2 = \emptyset$).*

Note that the disjoint-events-decidability condition means that we can also decide whether one event is a subset of another, since $e_1 \subseteq e_2$ is equivalent to $e_1 \cap (\Omega_{\mathbf{V}} \setminus e_2) = \emptyset$.

In the above condition we replaced the – in our view – too strict computable-measure condition with the disjoint-events-decidability condition. It is fulfilled for a wide class of possible ways to define events, e.g. linear inequalities on integers (excluding multiplication) [116] and inequalities on real numbers including multiplication [36]. Although we are not able to define arbitrary distributions, we can always define a set of distributions which includes any such distribution. For instance, queries of programs consisting of linear constraints on real numbers distributed according to arbitrary continuous distributions, can be approximated with known maximal error, as will be shown next. Additionally, it is possible to define imprecise distributions in case not enough knowledge is available about what the actual distribution is.

Example 4.17

Assume we have random variables $\mathbf{V}_1$ and $\mathbf{V}_2$ with the range of real numbers and the corresponding credal set specification:

$$\{\ (0.25, \omega_1 < 0 \land \omega_2 < 0), (0.25, \omega_1 < 0 \land \omega_2 > 0),\\ (0.25, \omega_1 > 0 \land \omega_2 < 0), (0.25, \omega_1 > 0 \land \omega_2 > 0)\ \}$$

Here we use a shorthand notation for events, only denoting the condition on elements of the sample space. This credal set specification for instance includes the case that ω_1 and ω_2 are independent and normally distributed with mean 0.0 and arbitrary standard deviation.

Suppose we want to compute the probability of the event $2\omega_1 > \omega_2$. We can decide that the following statements hold, since linear constraints on real-valued variables are decidable (disjoint-events-decidability condition):

$$(2\omega_1 > \omega_2) \not\supseteq (\omega_1 < 0 \land \omega_2 < 0)\\ (2\omega_1 > \omega_2) \not\supseteq (\omega_1 < 0 \land \omega_2 > 0)\\ (2\omega_1 > \omega_2) \supseteq (\omega_1 > 0 \land \omega_2 < 0)\\ (2\omega_1 > \omega_2) \not\supseteq (\omega_1 > 0 \land \omega_2 > 0)$$

From this we can compute $\underline{P}(2\omega_1 > \omega_2) = 0.25$. To determine the upper bound we observe the following:

$$(2\omega_1 > \omega_2) \cap (\omega_1 < 0 \land \omega_2 < 0) \neq \emptyset\\ (2\omega_1 > \omega_2) \cap (\omega_1 < 0 \land \omega_2 > 0) = \emptyset\\ (2\omega_1 > \omega_2) \cap (\omega_1 > 0 \land \omega_2 < 0) \neq \emptyset\\ (2\omega_1 > \omega_2) \cap (\omega_1 > 0 \land \omega_2 > 0) \neq \emptyset$$

From this we can compute $\overline{P}(2\omega_1 > \omega_2) = 0.75$.

4.5.2.5 *Approximating Continuous Distributions Using Credal Sets*

The examples in Section 4.3 contain continuous distributions, which would translate to infinite credal set specifications, for which therefore exact inference would not be possible. Credal sets can however be used to approximate combinations of arbitrary continuous distributions with known CDF. This is done by associating probabilities to intervals, defining a set of distributions including the actually intended one. To do that one can divide the sample space of a single variable $\mathbf{V}_i$ in n intervals $(l_1, u_1), \ldots, (l_n, u_n)$ where for all j: $l_j \leq u_j$, l_j may be a real number or $-\infty$ and u_i may be a real number or ∞. We can now define the following one-dimensional credal set:

$$\{P(l_1 < \mathbf{V}_i < u_1) \colon l_1 < \mathbf{V}_i < u_1, \ldots, P(l_n < \mathbf{V}_i < u_n) \colon l_n < \mathbf{V}_i < u_n\}$$

To compute probabilities of the integrals, we make use of the fact that one-dimensional integrals over typical PDFs can be computed with negligible error, for example for normal distributions such integrals can be computed using CDFs using the *error function*. The credal sets of independent random variables can be combined by taking the product of the credal sets, which means using all combinations of elements, taking the product of probabilities and the intersection of events. The same technique is also applicable for multivariate distributions by using *hyperrectangles* instead of intervals. For instance, in two dimensions ordinary rectangles and in three dimensions *rectangular cuboids* can be employed.

Providing a specification with more intervals restricts the possible distributions more, which means that one can get a credal set arbitrarily close to an arbitrary single distribution. However, since the credal set specification has to be finite, it generally cannot be restricted to an arbitrary single distribution. The probability bounds of each query can be used to determine the maximum error of the approximation. The number of intervals determine the precision, i.e. the gap between the probability bounds one can compute. So the precision can be increased arbitrarily. Figure 4.6 gives an example of a Gaussian distribution divided into five intervals with equal probability.

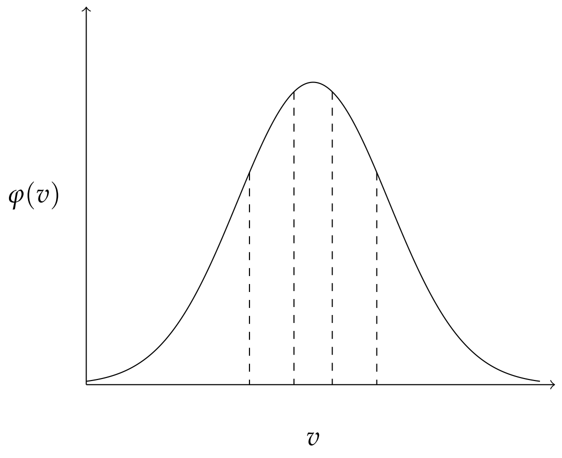


Figure 4.6: Discretised PDF of a Gaussian Distribution ($\varphi(v)$ is the density for the value v; the dashed lines indicate the intervals the distribution is discretised into.)

4.5.3 *PCLP Semantics*

This section introduces the concrete semantics of PCLP. The structure of rules is the same for the abstract semantics and PCLP. We therefore focus on how the abstract concept of credal set specifications is realised by the definitions as introduced in Section 4.4.

4.5.3.1 *Random Variable Definitions*

We start showing that a PCLP program defines a countable sequence of finite credal set specifications $\mathbf{C}_1, \ldots$ with the length of the number of random variables and therefore defines a credal set specification in the sense of Definition 4.6.

Definition 4.10 (Credal Set Specifications of a PCLP Program). *We associate a single random variable definition with each random variable, which is the first matching definition occurring in the program. If there is no matching definition we associate the specification* $\{1.0\colon true\}$*, which assigns the entire probability mass to the constraint which is always true, as this does not fix any information about the probability distribution. We fix the enumeration of random variables and denote the family of the definitions for the first n random variables as* $\mathbf{D}_n$*. From this we define the credal set specification concerning the first n random variables* $\mathbf{C}_n$ *as:*

$$\mathbf{C}_n \stackrel{\text{def}}{=} \left\{ \Big(p, \pi_n(\mathrm{CSS}(\varphi))\Big) \mid (p, \varphi) \in \hat{\prod_{d \in \mathbf{D}_n}} d \right\}$$

Here the product of two random variable definitions $d \hat{\times} e$ *is defined as:*

$$d \hat{\times} e \stackrel{\text{def}}{=} \{(p_d \cdot p_e, \varphi_d \wedge \varphi_e) \mid \big((p_d, \varphi_d), (p_e, \varphi_e)\big) \in d \times e\}$$

We take the product of the probabilities, since we assume them to be independent, and map the constraints to events by making use of their solution space (CSS). The projection to n dimensions (π_n) is necessary, because random variables not under the first n are possibly included in case they are defined together in the same definition with one of the first n.

Example 4.18

In Example 4.15 the two-dimensional credal set specification of the variables **Time_Comp**$_1$ and **Time_Comp**$_2$ was given as:

$$\begin{aligned}
\mathbf{C}_2 = \{ \; & \big(0.49, \{(\omega_1, \omega_2) \mid 0 \le \omega_1 \le 1, 0 \le \omega_2 \le 1\}\big), \\
& \big(0.14, \{(\omega_1, \omega_2) \mid 1 \le \omega_1 \le 2, 0 \le \omega_2 \le 1\}\big), \\
& \big(0.07, \{(\omega_1, \omega_2) \mid 2 \le \omega_1 \le 3, 0 \le \omega_2 \le 1\}\big), \\
& \big(0.14, \{(\omega_1, \omega_2) \mid 0 \le \omega_1 \le 1, 1 \le \omega_2 \le 2\}\big), \\
& \big(0.04, \{(\omega_1, \omega_2) \mid 1 \le \omega_1 \le 2, 1 \le \omega_2 \le 2\}\big), \\
& \big(0.02, \{(\omega_1, \omega_2) \mid 2 \le \omega_1 \le 3, 1 \le \omega_2 \le 2\}\big), \\
& \big(0.07, \{(\omega_1, \omega_2) \mid 0 \le \omega_1 \le 1, 2 \le \omega_2 \le 3\}\big), \\
& \big(0.02, \{(\omega_1, \omega_2) \mid 1 \le \omega_1 \le 2, 2 \le \omega_2 \le 3\}\big), \\
& \big(0.01, \{(\omega_1, \omega_2) \mid 2 \le \omega_1 \le 3, 2 \le \omega_2 \le 3\}\big) \; \}
\end{aligned}$$

This credal set specification can be represented in PCLP as:

$$\begin{aligned}\mathbf{Time\_Comp}_1 \sim \{\ & 0.7\colon 0 \leq \mathbf{Time\_Comp}_1 \leq 1,\\ & 0.2\colon 1 \leq \mathbf{Time\_Comp}_1 \leq 2,\\ & 0.1\colon 2 \leq \mathbf{Time\_Comp}_1 \leq 3\ \}\\ \mathbf{Time\_Comp}_2 \sim \{\ & 0.7\colon 0 \leq \mathbf{Time\_Comp}_2 \leq 1,\\ & 0.2\colon 1 \leq \mathbf{Time\_Comp}_2 \leq 2,\\ & 0.1\colon 2 \leq \mathbf{Time\_Comp}_2 \leq 3\ \}\end{aligned}$$

Lemma 4.4. *A PCLP program defines a credal set over all random variables $\mathbf{V}_1, \mathbf{V}_2, \ldots$ occurring in it.*

To derive formulas for the probability bounds of a PCLP program, we first introduce the concept of *solution constraint*, which is the equivalent of the solution event expressed in terms of constraints (cf. Lemma 4.2).

Definition 4.11 (Solution Constraint)**.** *The solution constraint of a query q is defined as:*

$$\mathrm{SC}(q) \stackrel{\text{def}}{=} \bigvee_{\substack{M_{\mathbf{L}}[\Phi] \models q\\ \Phi \subseteq \mathbf{Constr}}} \ \bigwedge_{\varphi \in \Phi} \varphi$$

Example 4.19

Consider again the clauses of the running example (Section 4.3.3):

$$\mathit{saved} \leftarrow \langle \mathbf{Time\_Comp}_1 < 0.75 \rangle$$
$$\mathit{saved} \leftarrow \langle \mathbf{Time\_Comp}_1 < 1.25 \rangle, \langle \mathbf{Time\_Comp}_1 + 0.25 \cdot \mathbf{Time\_Comp}_2 < 1.375 \rangle$$

The solution constraint for the query *saved* is:

$$\begin{aligned}\mathrm{SC}(\mathit{saved}) = {} & \mathbf{Time\_Comp}_1 < 0.75\\ & \vee (\mathbf{Time\_Comp}_1 < 1.25 \wedge \mathbf{Time\_Comp}_1 + 0.25 \cdot \mathbf{Time\_Comp}_2 < 1.375)\end{aligned}$$

The way to compute probabilities according to Proposition 4.2 requires deciding disjointness of events. We substitute this by deciding satisfiability of constraints. For this we introduce a function for checking satisfiability of constraints, which has to be filled in by a constraint checker for a concrete implementation. We also consider only partially decidable constraint theories. Non-linear constraints can for instance often be solved, but not in general.

Definition 4.12 (Satisfiability Check)**.** *The function*

$$\mathit{check}\colon \mathbf{Constr} \to \{\mathit{sat}, \mathit{unsat}, \mathit{unknown}\},$$

checks satisfiability of constraints. The possible values of the function have the following meaning:

- *sat:* *The constraint is certainly satisfiable, i.e. there is a solution.*
- *unsat:* *The constraint is certainly unsatisfiable, i.e. there is no solution.*
- *unknown:* *Satisfiability could not be decided, i.e. nothing is said about the constraint.*

In case the function never yields unknown for any constraint, we call the constraint theory fully decidable.

We can now express probability bounds of a query in a PCLP program using the satisfiability check function.

Proposition 4.4. *The lower and upper probability bounds of a query q, concerning only the first n random variables, in a PCLP program, fulfilling the exact inference conditions, can be computed as:*

$$\underline{P}(q) = \sum_{\substack{check(\varphi \wedge \neg \mathrm{SC}(q)) = unsat \\ (p,\varphi) \in \mathbf{C}_n}} p$$

$$\overline{P}(q) = \sum_{\substack{check(\varphi \wedge \mathrm{SC}(q)) = sat \\ (p,\varphi) \in \mathbf{C}_n}} p$$

Here $\mathbf{C}_n$ *are the credal set specifications defined by the program (Definition 4.10).*

Corollary 4.2. *For PCLP programs, making use of partially decidable constraints, the following holds:*

$$\underline{P}(q) \geq \sum_{\substack{check(\varphi \wedge \neg \mathrm{SC}(q)) = unsat \\ (p,\varphi) \in \mathbf{C}_n}} p$$

$$\overline{P}(q) \leq \sum_{\substack{check(\varphi \wedge \mathrm{SC}(q)) \neq unsat \\ (p,\varphi) \in \mathbf{C}_n}} p$$

Corollary 4.3. *Exact inference in* PCLP(**Constr**) *is possible for queries with a finite number (finite-relevant-constraints condition) of finite-dimensional constraints (finite-dimensional-constraints condition) and in case satisfiability can be decided for the constraint language* **Constr**.

4.5.3.2 *Inference Tasks*

PCLP can be used for a number of inference tasks as summarised by Table 4.2. In case we just have discrete, finite distributions we can compute the exact bounds of a query, which is a point probability for precise distributions, using Proposition 4.4.

In the hybrid case we are able to compute bounds using Proposition 4.4 as well. Such bounds are only approximations, since the random variable definitions we use are only approximations of the actual continuous distributions. For the precise case we know that the bounds approximate a point probability and therefore know the maximal error of the approximation. In the imprecise case the difference between the lower and upper bound is partially explained by approximated continuous distributions and partially by the imprecise immanent to the distributions. We therefore do not know how good the approximation is. In both cases the approximation asymptotically equals the actual result in case of infinite computation time.

Finally, in case we have a partially decidable constraint theory, we can still determine approximations using Corollary 4.2. The achievable precision may however be bounded by constraints which are not decidable.

	Precise: P(q)	**Imprecise**: $\underline{P}(q)$, $\overline{P}(q)$
Discrete	$P(q)$	$\underline{P}(q)$, $\overline{P}(q)$
Hybrid	$P(q) \pm \epsilon$ known, arbitrary ϵ	$\underline{P}(q) - \underline{\epsilon}$, $\overline{P}(q) + \overline{\epsilon}$ unknown, arbitrary $\underline{\epsilon}$, $\overline{\epsilon}$
Partially Decidable Constraints	$P(q) \pm \epsilon$ known, bounded ϵ	$\underline{P}(q) - \underline{\epsilon}$, $\overline{P}(q) + \overline{\epsilon}$ unknown, bounded $\underline{\epsilon}$, $\overline{\epsilon}$

Table 4.2: Overview of Inference Tasks

4.6 related work

There are various paradigms for probabilistic programming, which are related to non-probabilistic programming paradigms. They range from graphical formalisms (e.g. BNs [119], *multi-entity Bayesian networks* [84]), imperative and object-oriented languages (e.g. *FACTORIE* [94], *Figaro* [120]), a purely functional paradigm (e.g. *Church* [57]), LP languages (e.g. *independent choice logic* (ICL) [122], *ProbLog* [125]), to other logic-based languages (e.g. *BLOG* [102]). PCLP fits within the LP paradigm, which has been shown to be a powerful knowledge-representation tool for non-probabilistic problems. As a result, this is the first and arguably best studied first-order probabilistic programming paradigm, with a well-founded semantics.

In the following, rather than focusing on the underlying programming paradigm, we discuss the expressivity of related languages, i.e. which kinds of distributions can be represented. This is discussed along two dimensions: discrete versus hybrid distributions and precise versus imprecise distributions. The resulting classes of languages, together with some examples, are shown in Table 4.3 and will guide the rest of this section. As the table illustrates, being able to represent imprecise hybrid distributions is, to our knowledge, a unique feature of PCLP. Exceptions are the very general probabilistic logics [6, 63], discussed in Section 3.1, about which the work is however merely theoretical and does not provide a concrete inference method, let alone an implementation.

	Discrete	**Hybrid**
Precise	Bayesian Networks [119] ProbLog [125] PRISM [138] Independent Choice Logic [122]	Conditional Linear Gaussians [85] Hybrid Problog [60] Distributional Clauses [61] BLOG [102]
Imprecise	Locally Defined Credal Networks [31] Probabilistic Logic Programming [111] Imprecise Probabilistic Horn Clause Logic (Chapter 7)	Probabilistic Constraint Logic Programming

Table 4.3: Related Languages

4.6.1 *Precise Discrete Distributions*

PCLP clearly subsumes languages allowing one to define discrete and precise probability distributions including *propositional* ones, such as BNs, as well as first-order languages, e.g. probabilistic logics based on Sato's distributions semantics. Examples are ICL [122], PRISM [138], ProbLog [125] and CP-logic [152]. Languages in which weights instead of probabilities are used, such as *Markov logic networks* (MLNs) [44], have a very different nature compared to the above approaches, in the sense that the parameters in such models do not necessarily have a direct probabilistic meaning (cf. Section 3.3.2). A number of approaches use constraints to model discrete distributions. This is done with advantages for representation and learning in mind. There are a number of languages, which are, as PCLP, based on deterministic CLP. One example of such language is CLP($\mathcal{BN}$) [30], which does not use constraints to denote events and define probability distributions, but considers probability distributions as constraints themselves. The idea is quite different than the distribution semantics, as illustrated by some examples by Costa and Paes [29]. In particular, dynamic models, such as *Hidden Markov Models*, can be represented by PRISM very compactly and concisely, in contrast to CLP($\mathcal{BN}$). This observation similarly holds for other approaches based on the distributions semantics, including PCLP. An approach, similar to CLP($\mathcal{BN}$), but more general, is *clp(pdf(Y))* [2], which is however restricted to discrete random variables, too. Another languages by Riezler [127], sharing the name with PCLP, assigns probabilities to derivation choices. The languages is developed with natural language processing applications in mind and does not consider continuous distributions. Finally, Sato's CBPMs [140] are based on the idea that any discrete joint probability distribution can be expressed as independent binary random variables, constrained by arbitrary *first-order logic* (FOL) formulas. Constraints in this setting are therefore purely logical, without additional theories, and the method is restricted to discrete distributions, i.e. it cover at most countably infinite domains, such as integers.

A limitation of PCLP and most other probabilistic LP languages is that they do not directly support generative definitions, i.e. random variables cannot be defined in terms of the value of other random variables. For finite, discrete distributions, this can be circumvented by introducing distinct random variables, each related to a value of another one. However, this is not possible for infinite distributions. One example where generative definitions are useful is modelling an unknown number of objects, as shown by the work on BLOG [102]. The number of objects can for instance be modelled by a *Poisson distribution*, which is discrete, but not finite. Such generative processes could theoretically be modelled in PCLP, yet the number of relevant constraints would not be finite, and therefore exact inference may not be possible in PCLP (Theorem 4.3).

4.6.2 *Imprecise Discrete Distributions*

Formalisms dealing with imprecise probabilities can typically represent more complex imprecise distributions than PCLP, for example [149, 111]. Specifically, PCLP cannot express qualitative relations between probabilities of events, e.g. express that the probability of event a is greater than the probability of event b. However, the restricted way of how imprecise probabilities are defined in PCLP guarantees that definitions cannot be inconsistent (Theorem 4.1). Some languages that do guarantee consistency such as LDCNs [31] and relational variants [32] are still more expressive than PCLP, since they can express conditional credal sets. This comes with the price of increased inference complexity (cf. Section 5.3). PCLP can also not express open probability intervals, which is e.g. realised for *imprecise probabilistic Horn clause logic* (IPHL) (Chapter 7) by using infinitesimal probabilities.

4.6.3 *Precise Hybrid Distributions*

All approaches we discuss are not as general as PCLP, in the sense that continuous distributions are restricted to real-valued random variables. The semantic foundation of PCLP makes use of a more general notion of continuous distributions, concerning random variables with arbitrary uncountable ranges. There are in general roughly two different ways to deal with continuous distributions: computing posterior continuous distributions of continuous random variables or computing probabilities of events.

Examples of the first approaches are *conditional linear Gaussian* (CLG) models [85] or the related first-order approach in [71], which is based on the distribution semantics as well. The approach provides more information about the continuous distributions as possible with probabilities of events alone. This can for instance be used for computing expectations and decision making. However, the method requires a way to represent resulting PDFs of distributions. Those could be combinations of Gaussian distributions or other ways to represent functions as *mixtures of truncated basis functions* [83] and *piecewise polynomials* [136], which can approximate usually employed distributions, though no strong guarantees can be given that the approximation is close to the true distribution. Summarised, determining result distributions can only be achieved by severely restricting the distributions allowed (e.g. to CLG distributions) or by using an approximate representation.

The second approach to deal with continuous distributions – as adopted by PCLP – focuses on computing probabilities of events involving continuous variables. It is still powerful enough for decision making in case of discrete decisions. Within this category, most probabilistic languages based on imperative, object-oriented and functional approaches support hybrid distributions. A hybrid extension of a LP based language is Hybrid ProbLog [60], which allows real-valued random variables, but restricts their use to one dimensional constraints, for instance $\mathbf{X} > 0$. This framework is therefore less expressive than PCLP. One of the most general extensions of logical semantics are *distributional clauses* (DC) [61], which supports arbitrary constraints on real-valued random variables and generative definitions, e.g. the variance of a distribution could depend on the value of another one. Such generative definitions cannot be expressed by PCLP. However, incorporating generative definitions within the context of the distribution semantics complicates the semantics and puts a burden on the user of the language by requiring a number of very technical requirements for a program to be valid [61, Definition 3]. Our extension of the distribution semantics is – in our view – more close to the original idea and provides the necessary properties for our analysis of the exact inference conditions and our credal set extension.

4.7 CONCLUSIONS

We introduced PCLP, based on a generalised distribution semantics, making it possible to use random variables with arbitrary ranges. The semantics illustrate that the integration of various known techniques can provide a powerful formalism. In particular, we combine the ability of logic to represent relational knowledge, *probability theory* to deal with uncertainty and constraints representing conditions on variables with various ranges.

As commonly known, exact probabilistic inference is only possible under conditions which pose strong restrictions on distributions. As an alternative to many other approximation methods, we propose to use a framework based on credal sets where lower and upperbounds on posteriors can be guaranteed. Placing the method in the theory of imprecise probabilities provides a clear view on the approach, allowing one to explore its properties and to compare with other approaches. In particular, a unique property of PCLP compared to other probabilistic logic languages defining credal sets, is that it guarantees that definitions remain consistent, i.e. the credal set defined is guaranteed to contain at least one element. Also, powerful applications of the approach can straightforwardly be defined, for instance approximating probabilities of events with known and arbitrarily small maximum error in hybrid distributions.

5 INFERENCE BY GENERALISED WEIGHTED MODEL COUNTING

We have shown in the previous chapter that under certain conditions we can perform exact *inference* for *Probabilistic Constraint Logic Programming* (PCLP) in two steps: determining the *solution constraint* (Definition 4.11) and computing its probability using Proposition 4.4. Both steps have exponential time complexity if done in a naive way. The solution constraint can be determined by considering all subsets of constraints in the support set of the query. Then, probabilities can be computed by summing over all possible choices of which there are exponentially many in terms of the number of elements in *independent random variable definitions*.

Inference can often be done more efficiently by making use of the problem's structure. We show how this can be done for the first step (Section 5.1) and develop a novel algorithm for the second step (Section 5.2). We consider the theoretical properties of the proposed algorithm in terms of its complexity as key result for showing its potential and discuss those results in Section 5.3. We further experimentally evaluate our theoretical claims in Section 5.4.

5.1 DETERMINING THE SOLUTION CONSTRAINT

We only briefly describe how we derive the solution constraint, since it is a variation of existing techniques. The algorithm is based on *selective linear definite clause* (SLD) resolution, which is a well-known way to derive proofs of queries in *logic programming* (LP). We deal with constraints in bodies in the same way as the operational semantics of *constraint logic programming* (CLP) does: we collect all constraints we encounter during an SLD derivation [74]. A derivation proves the query in case all collected constraints are true, i.e. if the conjunction of these constraints is true. This means a query can be derived if the conjunction of constraints derived from at least one of the derivations is true. The solution constraint is therefore the disjunction of those conjunctions, which relates to the definition of solution constraints (Definition 4.11). The main advantage of SLD resolution is that it restricts the subsets of constraints that need to be considered to only those that prove the query. Note that this coincides with the way in which proofs are collected in the first implementation of *ProbLog* [125], where the constraints are *probabilistic facts* rather than general constraints.

Negation can be added in a straightforward way, but simple SLD resolution is not sufficient in case cycles are present. There are however solutions to the problem: either by translating cyclic rules to acyclic ones [76] or using tabled SLD resolution [91]. We do not go into detail here and assume it is possible to efficiently derive the solution constraint for each query. Possible optimisations applied for CLP, such as pruning derivations if the imposed constraints become inconsistent, are not discussed either.

5.2 COMPUTING PROBABILITIES

The proposed algorithm for computing the probability of a solution constraint, is a generalisation of probabilistic inference by *weighted model counting* (WMC). We first discuss this way of inference before introducing our generalisation.

5.2.1 *Probabilistic Inference by Weighted Model Counting*

Real problems often possess a lot of structure that can be exploited to speed up inference. Various inference methods have been developed to take advantage of certain kinds of structure. Examples are *variable elimination* [123] and *recursive conditioning* [34]. We focus on performing probabilistic inference by translation to a WMC problem. This has been shown to be an efficient inference method for *propositional* formalisms such as *Bayesian networks* (BNs) [22] and as well to be applicable to *probabilistic logics* based on the distributions semantics [48]. The approach exploits not only topological structure, but also local structure such as *determinism* [77] and *context-specific independence* [16].

The problem of model counting basically means to find the number of models of a propositional *knowledge base*. WMC is a generalisation of the problem, where each model has a weight. Those weights are defined in terms of weights attached to the literals. The weight of a model is then the product of the weights of all literals included in the model.

Efficient WMC algorithms are based on the observation that counts of components sharing no *atoms* can be computed independently [7]. Model counting then proceeds as follows. We assume the theory is expressed in *conjunctive normal form* (CNF). In case the disjunctions in the CNF can be split such that they share no atoms, weights can be computed independently and the weight of the entire CNF is the product of those weights; this step is referred to as *decomposition*. Otherwise, a *case distinction* has to be made for a single atom. Then one gets two CNFs: one with the assumption this atom is false and another one with the assumption it is true. The weight is then computed as the sum of the weights of both CNFs, which represent theories with disjoint models. It can happen that due to the structure of the problem, such as determinism, this choice leads to a theory which can be simplified and potentially includes fewer atoms or even becomes *true* or *false*, which means determinism in the problem can be exploited. The choice of atom order is essential for the efficiency of the algorithm. Different possible heuristics for this choice are for instance discussed by Sang et al. [134, Section 3.2].

Example 5.1

We illustrate with this example the basic idea of binary model counting using the algorithm explained above. The example will later be used to compare with the generalised WMC algorithm. We only show splitting and omit decomposition as the latter is identical in the generalised version.

We start with the solution constraint of Example 4.19 and convert it to CNF:

$$\begin{aligned}
&\mathrm{SC}(\mathit{saved}) = \\
&\quad \mathbf{Time\_Comp_1} < 0.75 \lor \\
&\quad (\mathbf{Time\_Comp_1} < 1.25 \land \mathbf{Time\_Comp_1} + 0.25 \cdot \mathbf{Time\_Comp_2} < 1.375) \\
&= \\
&\quad (\mathbf{Time\_Comp_1} < 0.75 \lor \mathbf{Time\_Comp_1} < 1.25) \land \\
&\quad (\mathbf{Time\_Comp_1} < 0.75 \lor \mathbf{Time\_Comp_1} + 0.25 \cdot \mathbf{Time\_Comp_2} < 1.375)
\end{aligned}$$

In the following, we abbreviate the primitive constraints as follows:

$$\begin{aligned}
\varphi_1 &= \mathbf{Time\_Comp_1} < 0.75 \\
\varphi_2 &= \mathbf{Time\_Comp_1} < 1.25 \\
\varphi_3 &= \mathbf{Time\_Comp_1} + 0.25 \cdot \mathbf{Time\_Comp_2} < 1.375
\end{aligned}$$

To illustrate the binary version in comparison with the generalised one, we further assume for now that φ_1, φ_2 and φ_3 represent independent binary *random variables*. WMC then proceeds by choosing an atom to split on, which means to continue with two branches with the assumption the atom is true and false respectively. This is illustrated in Figure 5.1.

Note that assuming the truth values of literals makes it always possible to immediately simplify the constraint, which is not the case in the generalised version as shown later. The computation stops in case *true* or *false* is derived.

Suppose φ_1, φ_2 and φ_3 have independent probabilities 0.1, 0.2 and 0.3 respectively. To compute probabilities we put the probability corresponding to the choice at each edge and replace *true* by 1.0 and *false* by 0.0, as shown in Figure 5.2. The probabilities of the remaining nodes are computed bottom-up by taking the sum of the probability associated to the children weighted by the probability associated to the edges. Then, the probability of the root node is the probability of the query q.

5.2.2 *Generalised Weighted Model Counting Involving Constraints*

For the general case, we no longer restrict to binary atoms. This makes deciding whether constraints have solutions more complex and for this we therefore employ so-called *satisfiability modulo theories* (SMT) solvers. The SMT problem

$$\begin{tikzcd}
 & (\varphi_1 \vee \varphi_2) \wedge (\varphi_1 \vee \varphi_3) \arrow[dl, swap, "\varphi_1"] \arrow[dr, "\neg\varphi_1"] & & \\
(\varphi_1 \vee \varphi_2) \wedge (\varphi_1 \vee \varphi_3) \mid \varphi_1 = \mathit{true} & & (\varphi_1 \vee \varphi_2) \wedge (\varphi_1 \vee \varphi_3) \mid \neg\varphi_1 = \varphi_2 \wedge \varphi_3 \arrow[dl, swap, "\varphi_2"] \arrow[dr, "\neg\varphi_2"] & \\
 & \varphi_2 \wedge \varphi_3 \mid \varphi_2 = \varphi_3 \arrow[dl, swap, "\varphi_3"] \arrow[dr, "\neg\varphi_3"] & & \varphi_2 \wedge \varphi_3 \mid \neg\varphi_2 = \mathit{false} \\
\varphi_3 \mid \varphi_3 = \mathit{true} & & \varphi_3 \mid \neg\varphi_3 = \mathit{false} &
\end{tikzcd}$$

Figure 5.1: Binary WMC Example (Deriving Possible Models)

is the problem of deciding whether a given *first-order logic* (FOL) theory including additional background theories is satisfiable. SMT solver technology can be used to efficiency decide whether two *events* are disjoint or not, i.e. whether the conjunction of the constraints representing them is satisfiable or not. Very efficient SMT solvers for e.g. linear arithmetic are currently available [39, 46]. Since we use those solvers without modifications, we do not discuss the algorithms employed to decide satisfiability.

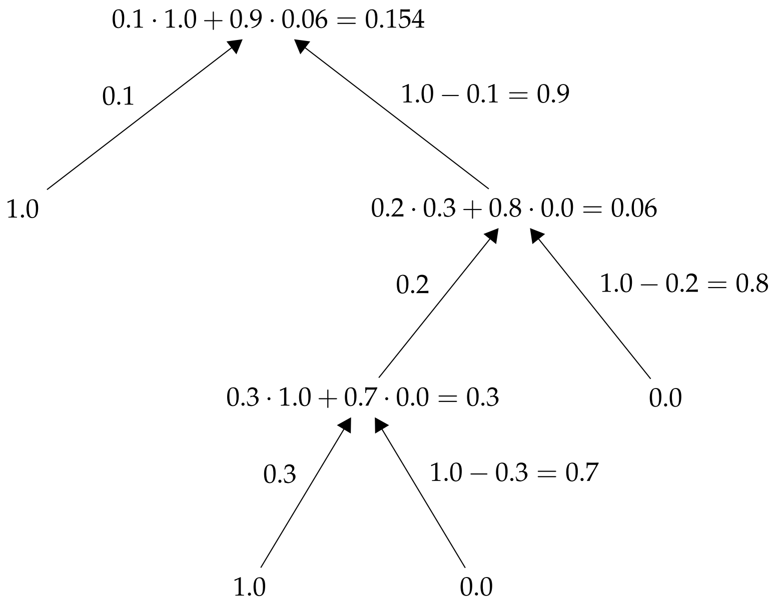
An SMT solver is used as an implementation of the function *check* (Definition 4.12). Even if the constraint theory is theoretically decidable, in practice a decision procedure might not be implemented. We refer to solvers which cannot decide satisfiability for all instances, i.e. for some cases return *unknown*, as *incomplete solvers*. Incomplete solvers have the same consequences for inference as discussed for partially undecidable constraint theories in Section 4.5.3.2.

As a way to implement elements of *credal set specifications* we, instead, make use of *choices* for the random variables, which are defined as follows.

Definition 5.1 (Choice). *A choice ψ is a function which selects for each random variable $\mathbf{V}_i$ a probability-constraint pair from its definition.*

Without loss of generality, we can restrict ourselves to those random variables that actually occur in the solution constraint of a query, as all other random variables have no influence on the probability of that query.

We generalise the two steps of WMC, which are case distinction and decomposition. The resulting algorithm (Algorithm 1) can be used to compute the bounds



$0.1 \cdot 1.0 + 0.9 \cdot 0.06 = 0.154$
0.1
$1.0 - 0.1 = 0.9$
1.0
$0.2 \cdot 0.3 + 0.8 \cdot 0.0 = 0.06$
0.2
$1.0 - 0.2 = 0.8$
$0.3 \cdot 1.0 + 0.7 \cdot 0.0 = 0.3$
0.0
0.3
$1.0 - 0.3 = 0.7$
1.0
0.0

Figure 5.2: Binary WMC Example (Computing Probabilities)

of a query's q probability as $\text{GWMC}(\text{SC}(q), \textit{true})$ and is discussed below. In case the SMT solver is incomplete we get bounds as given in Corollary 4.2.

Algorithm 1: Generalised Weighted Model Counting Algorithm (GWMC)

```
  Input: Constraint $\varphi$ in CNF and partial choice constraint $\Psi$
  Result: $(\underline{P}(\varphi \mid \Psi), \overline{P}(\varphi \mid \Psi))$
1 simplify $\varphi$ assuming $\Psi$
2 if $\varphi = \mathit{false}$ then return $(0.0, 0.0)$
3 else if $\varphi = \mathit{true}$ then return $(1.0, 1.0)$
4 else if there are independent CNFs $\varphi_1$, $\varphi_2$ in $\varphi$ then
5     return $\text{GWMC}(\varphi_1, \Psi) \cdot \text{GWMC}(\varphi_2, \Psi)$
6 else if there is some random variable $\mathbf{V}_i$ in $\varphi$, not occurring in $\Psi$ then
7     return $\sum_{\psi(\mathbf{V}_i)=(p,\Psi')} p \cdot \text{GWMC}(\varphi, \Psi \wedge \Psi')$
8 else return $(0.0, 1.0)$
```

5.2.2.1 *Case Distinction*

We first discuss how the process of distinguishing the cases that atoms are true or false is generalised. Instead of only two cases we have to consider all possible choices for the random variable at each level, as illustrated by the following example.

Example 5.2

We use the same solution constraint as in Example 5.1:

$$\begin{aligned}
&\mathrm{SC}(\mathit{saved}) = \\
&\quad(\mathbf{Time\_Comp}_1 < 0.75 \lor \mathbf{Time\_Comp}_1 < 1.25) \land \\
&\quad(\mathbf{Time\_Comp}_1 < 0.75 \lor \mathbf{Time\_Comp}_1 + 0.25 \cdot \mathbf{Time\_Comp}_2 < 1.375)
\end{aligned}$$

$$\begin{aligned}
\varphi_1 &= \mathbf{Time\_Comp}_1 < 0.75 \\
\varphi_2 &= \mathbf{Time\_Comp}_1 < 1.25 \\
\varphi_3 &= \mathbf{Time\_Comp}_1 + 0.25 \cdot \mathbf{Time\_Comp}_2 < 1.375
\end{aligned}$$

This time we treat φ_1, φ_2 and φ_3 as constraints instead of binary literals, which has as consequence that we cannot split on the two cases of the literal being true and false, but have to consider the choices given by the random variable definitions. We use the random variable definitions of Example 4.18:

$$\begin{aligned}
\mathbf{Time\_Comp}_1 \sim \{\ &0.7\colon 0 \le \mathbf{Time\_Comp}_1 \le 1, \\
&0.2\colon 1 \le \mathbf{Time\_Comp}_1 \le 2, \\
&0.1\colon 2 \le \mathbf{Time\_Comp}_1 \le 3\ \} \\
\mathbf{Time\_Comp}_2 \sim \{\ &0.7\colon 0 \le \mathbf{Time\_Comp}_2 \le 1, \\
&0.2\colon 1 \le \mathbf{Time\_Comp}_2 \le 2, \\
&0.1\colon 2 \le \mathbf{Time\_Comp}_2 \le 3\ \}
\end{aligned}$$

We denote the choices for $\mathbf{Time\_Comp}_1$ with ψ_{11}, ψ_{12} and ψ_{13} and the choices for $\mathbf{Time\_Comp}_2$ with ψ_{21}, ψ_{22} and ψ_{23}. We then at each level split on the choices of one variable as shown in Figure 5.3.

The important difference with binary WMC is that the constraint cannot always immediately be simplified. Simplification is only possible in case part of the constraint or its negation is a *consequence* of the choices. Therefore, for some branches choices for all random variables have been made, but still it cannot be decided whether the constraint is true or false. This corresponds to probability mass contributing to the upper, but not to the lower bound.

On the other hand, as for binary WMC, there are cases for which examining all choices is not necessary. For example, for the right-most branch beneath the root node in Figure 5.3, the choice ψ_{13} imposes $2 \le \mathbf{Time\_Comp}_1 \le 3$ which implies that φ_1 ($\mathbf{Time\_Comp}_1 < 0.75$) and φ_2 ($\mathbf{Time\_Comp}_1 < 1.25$) are false and therefore that the entire solution constraint is false. This shows that the order in which variables are chosen matters and – more importantly – that in this generalised setting we can still make use of determinism. The tree in Figure 5.3 would have 9 leaves if all choices were examined, but we can make use of the structure to reduce that to 7.

In general, if a subconstraint φ of the CNF is a consequence of the choices Ψ made so far ($\Psi \models \varphi$) it can be replaced by *true*. Similarly, if its negation is a

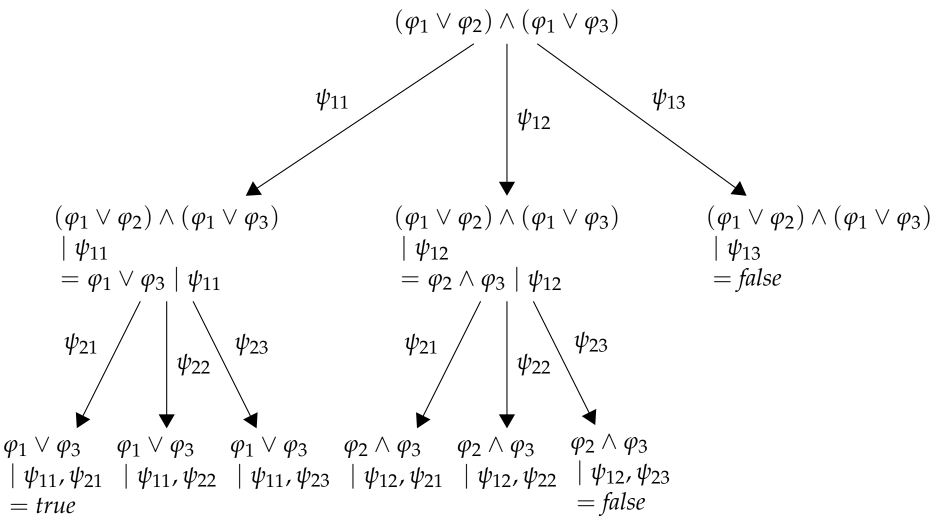

$(\varphi_1 \vee \varphi_2) \wedge (\varphi_1 \vee \varphi_3)$
ψ_{11}
ψ_{12}
ψ_{13}
$(\varphi_1 \vee \varphi_2) \wedge (\varphi_1 \vee \varphi_3) \mid \psi_{11} = \varphi_1 \vee \varphi_3 \mid \psi_{11}$
$(\varphi_1 \vee \varphi_2) \wedge (\varphi_1 \vee \varphi_3) \mid \psi_{12} = \varphi_2 \wedge \varphi_3 \mid \psi_{12}$
$(\varphi_1 \vee \varphi_2) \wedge (\varphi_1 \vee \varphi_3) \mid \psi_{13} = \mathit{false}$
ψ_{21}
ψ_{22}
ψ_{23}
ψ_{21}
ψ_{22}
ψ_{23}
$\varphi_1 \vee \varphi_3 \mid \psi_{11}, \psi_{21} = \mathit{true}$
$\varphi_1 \vee \varphi_3 \mid \psi_{11}, \psi_{22}$
$\varphi_1 \vee \varphi_3 \mid \psi_{11}, \psi_{23}$
$\varphi_2 \wedge \varphi_3 \mid \psi_{12}, \psi_{21}$
$\varphi_2 \wedge \varphi_3 \mid \psi_{12}, \psi_{22}$
$\varphi_2 \wedge \varphi_3 \mid \psi_{12}, \psi_{23} = \mathit{false}$

Figure 5.3: Generalised WMC Example

consequence of the choices ($\Psi \models \neg\varphi$), it can be replaced by *false*. The SMT solver can be used to check the conditions by making use of the fact that $\Psi \models \neg\varphi$ is a consequence of $\mathit{check}(\varphi \wedge \Psi) = \mathit{unsat}$ and $\Psi \models \varphi$ a consequence of $\mathit{check}(\neg\varphi \wedge \Psi) = \mathit{unsat}$. A subconstraint φ could for instance be a single primitive constraint in the CNF or the constraint represented by the entire CNF. After substitution the CNF can be simplified by the usual *logical* rules. The simplified CNF may enable further decomposition, since random variables may be removed.

To compute probability bounds we proceed as for binary WMC, except that we use probability pairs to represent the lower and upper bound. Leaves corresponding to *true* contribute to both bounds and are substituted by $(1.0, 1.0)$, leaves corresponding to *false* contribute to neither and are substituted by $(0.0, 0.0)$ and finally the leaves for which it is undecided contribute only to the upper bound and are substituted by $(0.0, 1.0)$. Each bound is then computed similar to the binary case.

Example 5.3

We continue computing probability bounds using the tree of Example 5.2, shown in Figure 5.3. In Figure 5.4 we illustrate how this computation is done. The result equals the bounds computed in Example 4.15, where we apply the semantic definition.

This idea is exploited in Algorithm 1 in Lines 1–3. We use Ψ to denote a partial choice, which is a conjunction of constraints chosen from random variable definitions, but possibly not from all relevant random variables yet. At Line 1 the current solution constraint is simplified as discussed above. Note that it is not efficient to always check all possible subconstraints at each step. There are

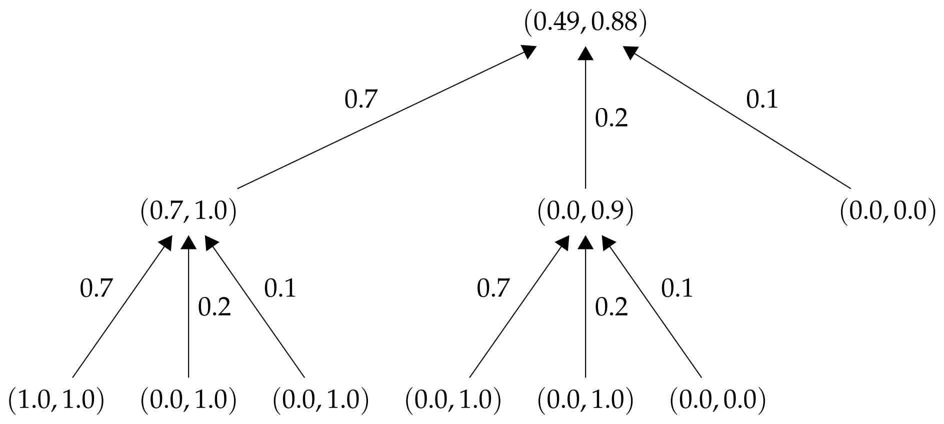


Figure 5.4: Generalised WMC Example (Computing Probabilities)

several strategies for which subconstraints to check, e.g. only checking primitive constraints for which all choices have been made and only checking the entire solution constraint at certain steps. In the implementation we use for the experiments we check all primitive constraints for which all choices have been made at each step. This strategy, in combination with logical simplifications, is often already sufficient to prove the solution constraint to be *false* or *true*. Only if this is not the case and all variables in the solution constraint have been chosen, our implementation checks the entire solution constraint.

If the constraint can be simplified to *true* or *false*, the branch of the computation is finished (Lines 2, 3). If a case distinction has to be made (Lines 6, 7) the efficiency of the further computations depends on the order variables are selected at Line 6. The main objective is to eliminate random variables to enable further decomposition, which is the same for binary WMC. In the generalised setting, the order of variables also determines how well determinism is exploited. As GWMC terminates in case an inconsistency can be found, a simple heuristic is to order random variables such that $\mathbf{V}_i < \mathbf{V}_j$ if $\mathbf{V}_i$ occurs in an (in)equality constraint with fewer variables on average than $\mathbf{V}_j$. For example, in the constraint $\mathbf{X} > 0 \wedge \mathbf{X} + \mathbf{Y} > 0$ we first select choices for $\mathbf{X}$ as some of these choices might make the whole constraint inconsistent.

In the implementation, which we use for the experiments, we prioritise exploiting independencies and use a simple counting heuristic for choosing variables. Concretely, we count the number of disjunctions in which a random variable occurs since, intuitively, eliminating variables occurring in more disjunctions gives more opportunities for decomposition. This heuristic is similar to the *dynamic largest combined sum* heuristic for the binary case [134, Section 3.2]. In case random variables have the same count we try to exploit determinism by choosing the variable occurring in a primitive constraint with the least number of other random variables together.

5.2.2.2 *Decomposition*

In the spirit of the well-known RelSAT algorithm [7] for WMC, we can also observe that in many cases the problem can be decomposed into subconstraints which do not share any random variables.

Example 5.4

Consider the solution constraint $\mathbf{V}_1 > 0 \wedge \mathbf{V}_2 > 0$ and let $|\Psi_{\mathbf{V}_i}|$ denote the number of choices for the random variable $\mathbf{V}_i$. In this case $\overline{P}(\mathbf{V}_1 \wedge \mathbf{V}_2) = \overline{P}(\mathbf{V}_1 > 0) \cdot \overline{P}(\mathbf{V}_2 > 0)$, which can be computed by examining $|\Psi_{\mathbf{V}_1}| + |\Psi_{\mathbf{V}_2}|$ choices only, whereas naively $|\Psi_{\mathbf{V}_1}| \cdot |\Psi_{\mathbf{V}_2}|$ would have to be examined. The same is true for the lower bound.

In Algorithm 1 we decompose if possible (Line 4) and recursively compute the bounds for both CNFs (Line 5).

5.2.2.3 *Other Optimisations*

We discuss some further optimisation, resulting in the refined Algorithm 2. Some optimisations for binary WMC can straightforwardly be generalised. Consider for example caching. For binary WMC, in case the same CNF is encountered twice, it has the same count. So computing such subproblems multiple times can be avoided using caching. In our generalised setting the CNF may still contain random variables for which a choice is already made, which means the same CNF can have different counts. We can, however, still make use of caching by reusing results for equal CNFs in combination with equal choices made for all random variables occurring in the CNF. So we first check whether a result is already cached (Line 3), otherwise we compute the result and store it in the cache (Line 17).

The characteristics of our generalised problem require additional optimisations. The fact that after making choices for random variables, the truthfulness of constraints involving them may still be undetermined, can dramatically hinder opportunities for decomposition. To counter this we implemented two optimisations eliminating such undetermined constraints. First, in case all variables occurring in a disjunction of the CNF are chosen, but the disjunction is still undetermined, the lower bound of the CNF is 0.0: even if all other components of the CNF will reduce to *true*, this single disjunction will lead to an undetermined CNF. Conversely, for determining the upper bound we assume the disjunction holds, which is realised by adding it to the constraints imposed by the choices, as formalised in Lines 6–8 of Algorithm 2. This leads to elimination of the disjunction and, therefore, a simplified CNF. It can be easily shown that the upper bound for the original CNF equals the upper bound for the simplified CNF with the additional assumption included in Ψ. Secondly, in case all disjunctions in the CNF have an undetermined constraint in common, it follows that the upper bound is 1.0. This is because the constraint alone can make the solution constraint true and the common constraint will never proven to be false, as the

Algorithm 2: Optimised Generalised Weighted Model Counting Algorithm (OGWMC)

```
1  Assuming an initially empty global cache map: cache
   Input: Constraint $\varphi$ in CNF and partial choice constraint $\Psi$
   Result: $(\underline{P}(\varphi \mid \Psi), \overline{P}(\varphi \mid \Psi))$
2  simplify $\varphi$ assuming $\Psi$
3  if      $(\varphi, \Psi)$ in cache then return $cache(\varphi, \Psi)$
4  else if $\varphi = false$       then return $(0.0, 0.0)$
5  else if $\varphi = true$        then return $(1.0, 1.0)$
6  else if there is a disjunction $\varphi'$ in $\varphi$ and all variables in $\varphi'$ also occur in $\Psi$ then
7      $lower, upper = \mathrm{OGWMC}(\varphi, \Psi \wedge \varphi')$
8      $result \quad = (0.0, upper)$
9  else if there is a primitive constraint $\varphi'$ occurring in all components of $\varphi$ and all
   variables occurring in $\varphi'$ occur in $\Psi$ as well then
10     $lower, upper = \mathrm{OGWMC}(\varphi, \Psi \wedge \neg\varphi')$
11     $result \quad = (lower, 1.0)$
12 else if there are independent CNFs $\varphi_1$, $\varphi_2$ in $\varphi$ then
13     $result = \mathrm{OGWMC}(\varphi_1, \Psi) \cdot \mathrm{OGWMC}(\varphi_2, \Psi)$
14 else if there is some random variable $\mathbf{V}_i$ in $\varphi$, not occurring in $\Psi$ then
15     $result = \sum_{\psi(\mathbf{V}_i)=(p,\Psi')} p \cdot \mathrm{OGWMC}(\varphi, \Psi \wedge \Psi')$
16 else return $(0.0, 1.0)$
17 $cache(\varphi, \Psi) \hookleftarrow result$
18 return result
```

choices for all variables occurring in it are already made. In this case, the lower bound equals the lower bound of the CNF assuming the negation of the constraint. This is formalised in Lines 9–11 of Algorithm 2.

5.3 COMPLEXITY ANALYSIS

To obtain insight into the complexity of the inference problem of PCLP, we analyse the main source of computational complexity, which is the computation of probabilities from a solution constraint. Very often, *imprecise* probabilistic inference is in general much more complex than precise probabilistic inference. For instance, obtaining a bound on a *marginal* probability given a *locally defined credal network* (LDCN) is $\mathsf{NP}^{\mathsf{PP}}$-complete, while it is PP-complete for BNs [37]. In order to compare the complexity of PCLP to LDCNs, we first show that the decision problem associated with PCLP inference is similar to BN inference for most practical constraint theories, i.e. it is in PP rather than $\mathsf{NP}^{\mathsf{PP}}$-hard. This reduced complexity comes at the cost of expressiveness, as discussed in Section 4.6.2. Subsequently, it is proven that PCLP also has complexity properties similar to BN inference and WMC if the *treewidth* of the problem is bounded.

5.3.1 *Worst-case Complexity*

By Corollary 4.1 it is clear that computing the lower and upper bound of a PCLP query has the same complexity. Furthermore, we turn the problem into a decision problem to ease complexity analysis, as is commonly done in the context of analysing the complexity of probabilistic inference. Therefore, the problem we consider is, given the solution constraint of a query q and any probability p, whether $\underline{P}(q) > p$. We first show that, given the complexity of constraint checking C, this problem is in PP^{C}.

Theorem 5.1. *Given a PCLP program* **Prog** *and a solution constraint φ for a particular query q, determining a bound on $\underline{P}(q)$ is in PP^{C}, where C is the complexity of checking satisfiability of sets of constraints in* **Prog***.*

If the complexity of SMT solving is polynomial, i.e. $\mathsf{C} = \mathsf{P}$, then the problem is in PP. Furthermore, proving PP-hardness is trivial as MAJSAT (deciding whether at least half of assignments to a propositional formula satisfy it) is PP-complete and can obviously be reduced to bound on the probability of a solution constraint. This shows that the main PCLP inference problem is as complex as precise, discrete probabilistic inference if checking constraints is tractable, even though we can deal with random variables of arbitrary domains and perform a particular kind of imprecise inference.

In the following, we will abstract from the complexity of SMT solving. For many interesting constraint domains such as equalities for discrete constraints and inequalities on sums and differences of real numbers, the complexity of SMT solving is polynomial [45, 79]. In other theories it is often desirable to use incomplete solvers, e.g. polynomial algorithms for theories where the satisfiability problem is NP-complete (e.g. arithmetic with integers [116]), even harder (e.g. real number arithmetic including multiplication [36]) or undecidable in general (e.g. arithmetic with integers including multiplication [93]). Finally, note that for typical problems the size of constraints is small compared to the size of the entire probabilistic inference problem.

5.3.2 *Parametrised Complexity in Terms of Treewidth*

Inference algorithms exploit the structure of problems. For example, the upper bounds of inference in BNs can more precisely be expressed in terms of the network's treewidth t [13], as $O(c^t)$, where c is the maximum cardinality of the variables. Instead of the cardinality of variables, we use the maximum number of choices d in the context of PCLP. The number of choices d is bounded by 2^{c^n}, where n is the number of variables. It is however usually much smaller, which is the strength of PCLP, especially in case c is infinitely large.

The treewidth measures how tree-like a graph is. In case a BN is a tree, inference can be done in linear time. A formal definition of treewidth is given below. In our context, we want to bound complexity in terms of a property of the input CNF. There are different kinds of graphs representing the structure of CNFs. For different kinds of those graphs, algorithms for determining the model count of CNFs are known, of which the complexity is bounded in terms of the graph's treewidth [132]. We here focus on the *primal graph*, which is the undirected graph with all CNF's atoms as nodes in which all nodes occurring together in a disjunction of the CNF are connected.

For the PCLP inference problem we slightly adapt the definition of a primal graph. First, we have to use random variables instead of atoms. Secondly, we also have to consider additional dependencies between random variables. Those dependencies emerge from the fact that it may not be possible to eliminate constraints, in spite of the fact that choices for all random variables occurring in them have been made, as discussed in Section 5.2.2.

We refer to *imprecise constraints* as constraints φ for which there is a choice Ψ for all variables occurring in φ, which cannot be used to simplify φ. That is neither $\Psi \models \varphi$ nor $\Psi \models \neg\varphi$.

Definition 5.2 (Constraint Primal Graph)**.** *The* constraint primal graph *of a CNF with constraints as leaves, is the undirected graph with all random variables occurring in the CNF as nodes. Two nodes* $\mathbf{X}$ *and* $\mathbf{Y}$ *are connected if and only if one of the following conditions hold:*

1. $\mathbf{X}$ *and* $\mathbf{Y}$ *occur together in a disjunction of the CNF.*
2. $\mathbf{X}$ *occurs together in a disjunction with an imprecise constraint* φ*,* $\mathbf{Y}$ *occurs together in a disjunction with an imprecise constraint* χ*, and* φ *and* χ *share random variables.*

Example 5.5

As an example consider the following CNF:

$$(\mathbf{X} > 0 \lor \mathbf{Y} > 0) \land \mathbf{Y} > \mathbf{Z}$$

The constraint primal graph for the case none of the constraints is imprecise is given in Figure 5.5a. In case the constraint $\mathbf{Y} > \mathbf{Z}$ is imprecise, an additional edge has to be added as shown in Figure 5.5b.

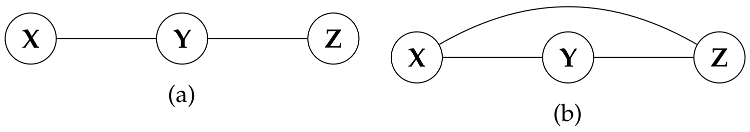


Figure 5.5: Example Constrained Primal Graphs

To measure the complexity of the constraint primal graph, we introduce the existing notions of *tree decomposition* and treewidth.

Definition 5.3 (Tree Decomposition [43]). *A tree decomposition of an undirected graph $G = (V, E)$ is a tree $T = (W, H)$, with nodes $x_1, \dots, x_n$, where each x_i is a subset of V, satisfying:*

1. *T includes only and all nodes of G.*
2. *All nodes connected by edges in G occur together in at least one node of T.*
3. *If two vertices of T contain a vertex of G, then all nodes of the tree in the path between those vertexes contain this vertex of G as well.*

Definition 5.4 (Treewidth [43]). *The treewidth of a graph is the width of its tree decomposition with minimal width. The width of a tree decomposition is the size of its largest node minus 1.*

Example 5.6

A tree decomposition with minimal width for the graph shown in Figure 5.5a, is given in Figure 5.6a. Therefore, the treewidth of this graph is 1. For the graph of Figure 5.5b only one tree decomposition is possible, shown in Figure 5.6b. The treewidth is increased to 2.

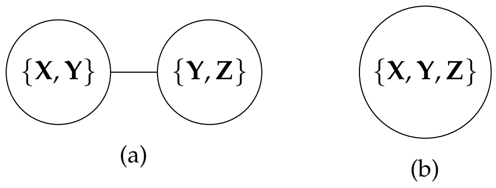


Figure 5.6: Example Tree Decompositions

Determining the treewidth of a given graph is NP-complete [4]. However, for a fixed treewidth a tree decomposition can be constructed in linear time [12].

Finally, we present the main result of this section, which relates the treewidth of the input CNF's constraint primal graph to the complexity of inference.

Theorem 5.2. *Given an input CNF of bounded treewidth t, the complexity of Algorithm 2 with proper variable order is $O(m\,t\,d^t)$, ignoring constraint checking, where m is the number of nodes of the tree decomposition and d is the maximal number of choices for a single random variable.*

We emphasize that this complexity is similar to the results for ordinary model counting, for instance the complexity in terms of the primal graph found by Samer and Szeider is $O(o\,m\,t\,2^t)$, where o is the maximum number of occurrences over all variables [132].

5.3.3 *Making use of Additional Structure*

Chavira et al. [22] argue that standard algorithms have complexity $\Theta(c^t)$, i.e. not only the worst-case, but as well the best-case complexity, is bounded exponentially by the treewidth. In contrast, probabilistic inference based on WMC can make use local structure, such as determinism [77] and context-specific independence [16], and can therefore in some cases perform better. Superiority of WMC-based inference has experimentally been shown as well [135]. The same is true for our generalised WMC algorithm. Example 5.2 for instance shows that branches can be pruned, which means not all exponentially many choices have to be examined. This is further evaluated experimentally in the following.

5.4 EXPERIMENTS

In this section, we provide some insight into the behaviour of the proposed algorithm. In particular, we show that our inference algorithm is competitive with existing approaches for discrete problems, i.e. the overhead of handling constraints and computing bounds instead of single probabilities is negligible. We further investigate scalability of the algorithm and show that our algorithm can make use of determinism. The implementation is available at `http://www.steffen-michels.de/pclp`. It makes use of *YAP Prolog* 6.2.2 and the SMT solver YICES 2.0.1 [46], which supports linear arithmetic. The experiments are run under Ubuntu 14.04 on a Laptop with an Intel Core i3 2.4 GHz processor and 4GB of RAM. All runtimes are averaged over 10 runs.

5.4.1 *Comparison with Other Implementations*

We compare our PCLP implementation with implementations based on similar inference algorithms. Such implementations are limited to precise distributions and mostly to discrete variables. We therefore use a precise, discrete problem to compare. The question we want to answer with the experiment is how scalability of our generalised inference algorithm compares to existing approaches. We expect some overhead for computing bounds instead of point probabilities and handling constraints. Note that, in this case computing bounds is superfluous, as the bounds will be equal. However, scalability of the generalised algorithm should be similar.

We compare to ProbLog 1 [125] and ProbLog 2 [48]. The first version of ProbLog works similar to our implementation. First, all explanations are collected using SLD resolution, resulting in a formula similar to what we call the solution constraint. The models of such formula are then counted by compilation to *binary-decision diagrams* (BDDs) [18]. ProbLog 2 uses a different approach. All grounded rules are translated to equivalent propositional formulas, which defines the WMC problem. This can yield more compact CNFs, but requires inclusion of all head atoms in the CNF, instead of only the probabilistic facts, as in the formerly mentioned proof-based approach. The CNF is then compiled to *deterministic decomposable negation normal form* (d-DNNF) [33]. Because the performance of this approach mainly depends on the d-DNNF compiler used, we provide results for two difference compilers: *C2D* [35] and the *DSHARP* compiler [106]. A fair comparison with *PRISM* is not possible, as it makes the strong assumption that bodies of the same head are exclusive. It therefore does not solve the difficult problem of computing probabilities of possibly overlapping bodies, according to the *inclusion-exclusion principle*, as the formerly mentioned approaches do.

For the experiment we use a version of the fruit selling problem, which we introduced in Section 4.3.1. In the original version we compare the actual price with the price customers are willing to pay:

$$buy(Fruit) \leftarrow price(Fruit, P),\ \langle P \leq \mathbf{Max\_price}(Fruit)\rangle$$

We replace this with discrete random variables, indicating whether the customer is willing to pay the price with and without government support:

$$\begin{aligned}\mathbf{Buys\_with\_support}(Fruit) &\sim \{0.3\colon yes, 0.7\colon no\}\\ \mathbf{Buys\_without\_support}(Fruit) &\sim \{0.6\colon yes, 0.4\colon no\}\end{aligned}$$

The definition of *buy* becomes then:

$$\begin{aligned}buy(Fruit) &\leftarrow \langle \mathbf{Support}(Fruit) = yes\rangle,\ \langle \mathbf{Buys\_with\_support}(Fruit) = yes\rangle\\ buy(Fruit) &\leftarrow \langle \mathbf{Support}(Fruit) = no\rangle,\ \langle \mathbf{Buys\_without\_support}(Fruit) = yes\rangle\end{aligned}$$

In the experiment we query $buy(fruit_1) \vee \cdots \vee buy(fruit_n)$ with increasing n and evaluate inference time. Ideally, inference time should scale linearly, since the query is a disjunction of n independent formulas. This however requires an efficient caching and elimination mechanism.

The result is shown in Figure 5.7. It clearly shows that PCLP is competitive, in spite of the overhead of calling the SMT solver and computing bounds instead of point probabilities. However, we do not achieve a linear complexity as would be ideally expected. Only *C2D* scales linearly; it is superior for very large n (not plotted in the graph), although for small n all other implementations perform better.

We adapt the problem and introduce a continuous random variable for the fruit's price. The price the customer is willing to pay is still not uncertain, which makes the constraint one-dimensional. This means it can be dealt with by *Hybrid ProbLog* [60], which is integrated in the ProbLog 1 implementation we used in the former experiment. For our PCLP implementation we discretise continuous distributions at the points of those constants, which guarantees a precise distribution. ProbLog 2 cannot deal with the problem, so we cannot include it in this comparison.

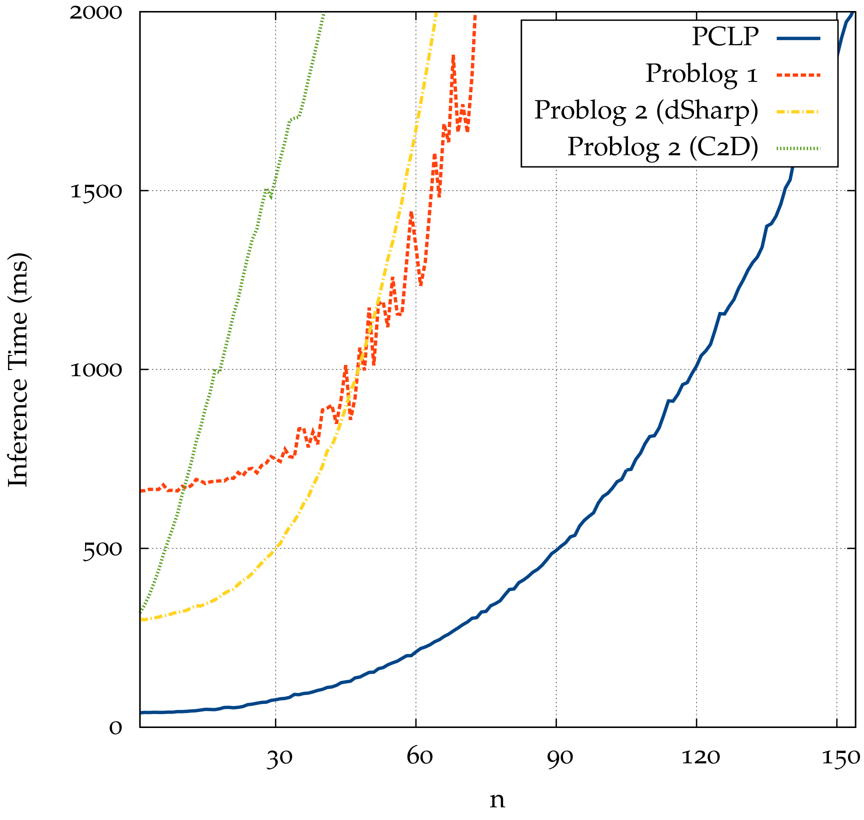


Figure 5.7: Comparison of PCLP with other Implementations for a Discrete Problem

The price of a kind of fruit is defined as in the example in Section 4.3.1:

$$\begin{aligned}
&\mathbf{Yield}(Fruit) &&\sim \mathcal{N}(12\,000.0, 1000.0)\\
&\mathbf{Support}(Fruit) &&\sim \{0.3\colon yes, 0.7\colon no\}
\end{aligned}$$

$$basic\_price(Fruit, 250 - 0.007 \cdot \mathbf{Yield}(Fruit))$$

$$\begin{aligned}
&price(Fruit, BPrice + 50) &&\leftarrow basic\_price(Fruit, BPrice),\ \langle \mathbf{Support}(Fruit) = yes\rangle\\
&price(Fruit, BPrice) &&\leftarrow basic\_price(Fruit, BPrice),\ \langle \mathbf{Support}(Fruit) = no\rangle
\end{aligned}$$

For the price customers are willing to pay we just use a constant. The rule defining *buy* becomes then:

$$buy(Fruit) \leftarrow price(Fruit, P),\ \langle P \leq 100.0\rangle$$

We again evaluated inference time for the query $buy(fruit_1) \vee \cdots \vee buy(fruit_n)$ with increasing n. The result in Figure 5.8 shows again that the performance of PCLP is competitive. The fact that PCLP is superior confirms the better results in the previous experiment. As the time for handling continuous distributions is negligible for this problem, the result reflects the higher overhead of the BDD package, used by *ProbLog 1*.

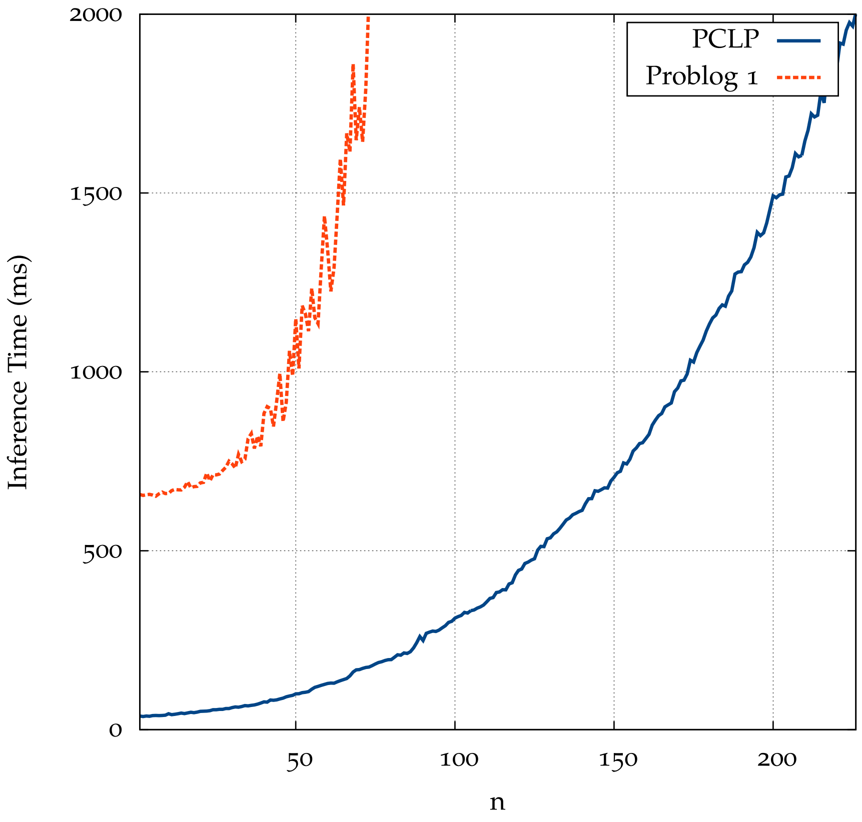


Figure 5.8: Comparison of PCLP with Hybrid ProbLog for a Continuous Problem

5.4.2 *Scalability*

We now consider the original problem as in Section 4.3.1, which means we use the following rule to model whether customers buy a certain kind of fruit:

$$buy(Fruit) \leftarrow price(Fruit, P), \langle P \leq \mathbf{Max\_price}(Fruit) \rangle$$

Inference for this problem cannot be performed by any of the frameworks we compared to previously, and to our knowledge PCLP is the only framework that can provide approximations with known error. To perform inference with PCLP, we discretise continuous distributions as described in Section 4.5.2.5 by naively choosing n intervals with equal probability, using inverse *cumulative distribution functions* (CDFs). In the experiments we increase n to decrease the maximal error of the approximation, which increases inference time. Note that this naive discretisation scheme is used on purpose, since we focus on evaluating the generalised WMC algorithm and showing its potential. Much better performance can be achieved by more sophisticated discretisation, as will be shown in Chapter 6.

As before we compute $P(buy(fruit_1) \vee \cdots \vee buy(fruit_n))$, but this time determine approximations. The result is shown in Figure 5.9, where we show the relationship between inference time and maximum error for varying n.

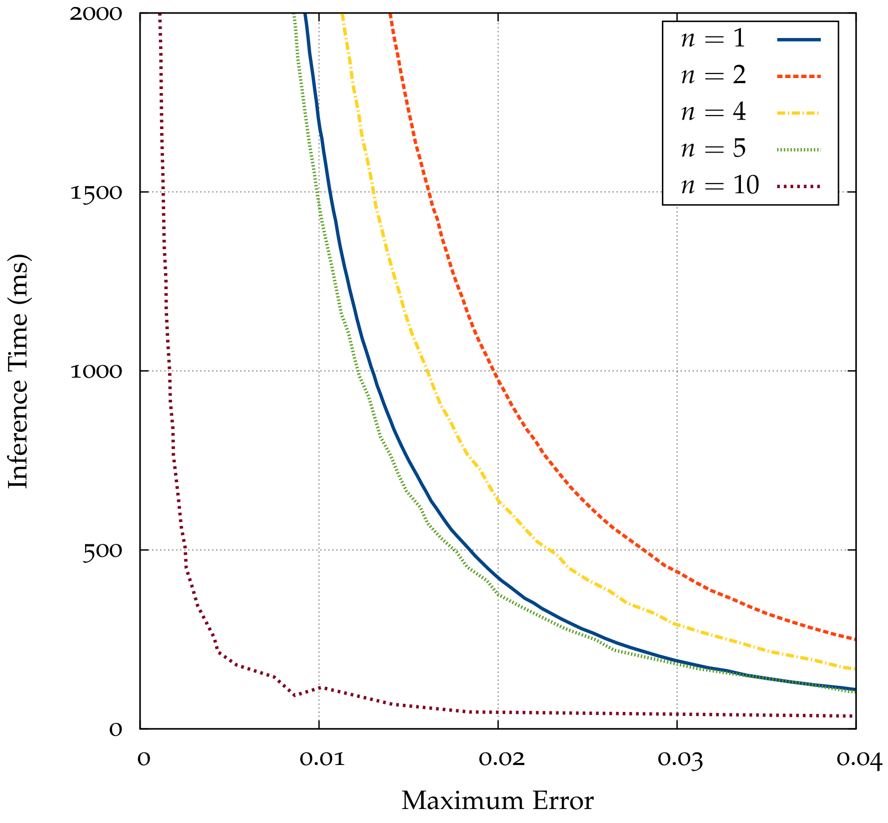


Figure 5.9: Inference Time vs Maximum Error for Query $buy(fruit_1) \vee \cdots \vee buy(fruit_n)$

The result shows an interesting non-monotonic behaviour: while for $n = 2$, inference is more expensive than for $n = 1$, for larger n the efficiency of inference increases with increasing n. For $n = 10$ we achieve a substantially better performance than for small n.

The efficiency decreases between $n = 1$ and $n = 2$, because in general in higher dimensional spaces, a finer discretisation is necessary to achieve the same precision. So with a naive inference approach, the maximum error would always be higher with larger n given the same inference time. However, by making use of the problem's structure the inference time is sub-exponential in the size of the choice space, in particular because subconstraints are independent. That effect dominates for larger n and the error can even decrease given the same inference time.

The experiment shows the potential of this inference method. Due to the structure of problems, often sub-problems concerning continuous variables only have to be estimated with low precision in order to achieve a satisfiable precision for the entire problem.

5.4.3 *Exploiting Determinism*

We finally show that the inference algorithm can make use of determinism by comparing inference times of the query $buy(apple) \wedge \langle \mathbf{Crop}(apple) > X \rangle$ for various X. The topological structure of the problems is the same, but with larger X the query can already be determined to be unsatisfiable, only based on the choice for the random variable **Crop**(*apple*). The result is shown in Figure 5.10, where we show again the relationship between inference time and maximum error.

For $X = 10\,000$ the result is similar to the result for the query $buy(apple)$, as the probability of the excluded subspace $\mathbf{Crop}(apple) \leq 10\,000$ is very small. With higher X however, the inference efficiency drastically increases, i.e. to achieve a certain precision less inference time is needed. The experiment confirms that the inference algorithm exploits determinism, similar to WMC algorithms in the binary case.

5.5 RELATED WORK

In this subsection, we will consider related methods for probabilistic inference, subdivided by exact, approximate, and lifted inference.

5.5.1 *Exact Inference*

There are various algorithms for exact probabilistic inference, most of them designed for BNs. Examples are variable elimination [123] and recursive conditioning [34]. As discussed, we base our work on the method of translating inference to WMC problems [22], since this method makes it possible to exploit struc-

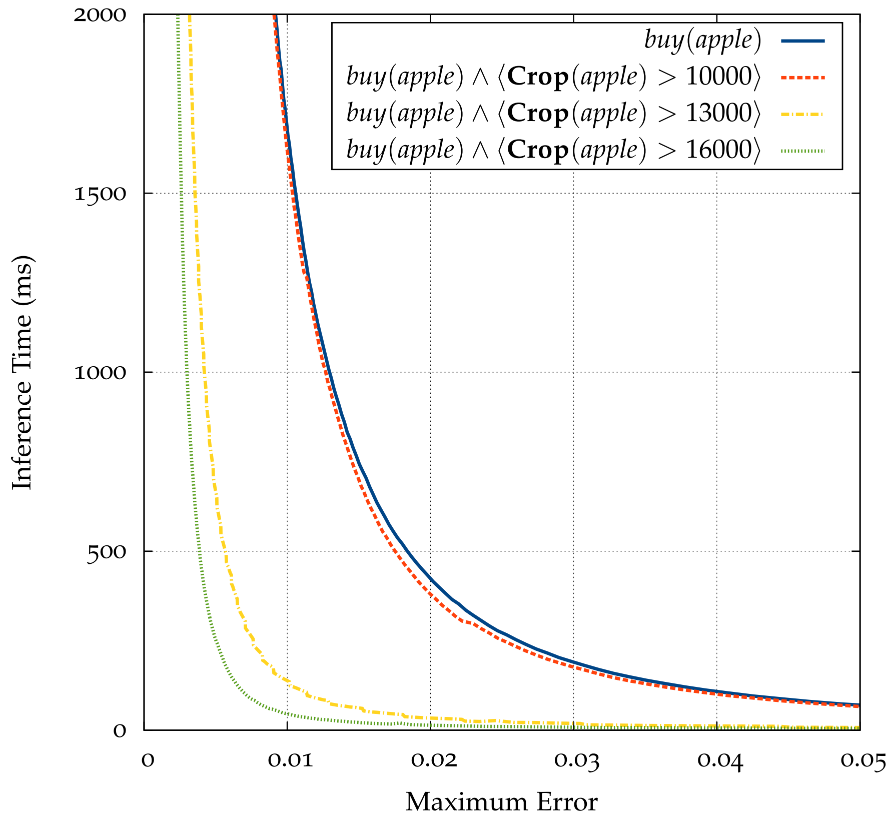


Figure 5.10: Inference Time vs Maximum Error for Query $buy(apple) \wedge \langle \mathbf{Crop}(apple) > X\rangle$ with Varying X

ture as determinism [77] and context-specific independence [16], which often occurs in logical theories. Exact inference is also possible for hybrid models with restricted types of distributions, for example *conditional linear Gaussian* (CLG) models [85]. Another hybrid formalism for which exact inference is possible is Hybrid ProbLog [60]. Since in this language constraints are one-dimensional, distributions can be discretised at all points occurring in inequalities in the proofs of a query, which turns inference into a discrete problem.

A more general related approach to inference with probabilities and constraints is by Dechter [42], which focuses on algorithms for various inference problems. All considered problems share the property that the structure of problem instances can be represented as graphs. Examples of such problems include probabilistic reasoning and constraint satisfiability, which are exactly the problem also unified in our approach. A particular algorithm for performing probabilistic inference modulo theories is presented in [41]. It is closely related to our method, but restricted to exact inference. Still this work may be the basis of interleaving probabilistic reasoning and constraint checking in a more efficient way in PCLP inference, based on a uniform representation combining both aspect of the inference problem. In our current approach, the probabilistic part of inference only uses the information which random variables occur in primitive constraints, but does not make use of the structure of constraints.

Exact inference algorithms have also been developed for the imprecise case. Since these problems are often strictly more complex than the precise case, methods for the precise case are not applicable. For instance, one has to resort to methods as *multi-linear programming* [32]. An exception is *imprecise probabilistic Horn clause logic* (IPHL), introduced later in this thesis (Chapter 7), providing an imprecise formalism for which complexity is as hard as for the precise case. However, IPHL restricts the language to binary random variables and *Horn clauses*, whereas PCLP does not impose such restrictions.

5.5.2 *Approximate Inference*

For BNs one class of approximation algorithms is based on Pearl's *belief propagation* algorithm [119], which computes posterior probabilities by passing messages between nodes. However, the algorithm only terminates and produces correct probabilities for networks which are polytrees, for which inference can be done in polynomial time. For the general case, the algorithm also often converges to a good approximation of the correct probabilities. A lot of work has been done about convergence and quality of such approximations [107, 159, 146, 105], but hard guarantees about the quality of the result cannot always be provided, in contrast to our inference method.

Similarly to PCLP, there is work on approximation schemes providing hard guarantees for discrete distributions by simplifying the problem [121]. Recently, the effectiveness of this kind of method in combination with state-of-the-art knowledge compilation techniques has been shown [128, 129, 153]. The main difference between PCLP and this work is that PCLP can be used to model and reason with continuous and imprecise distributions, whereas the former approaches are aimed at approximate inference for discrete and precise problems.

For continuous distributions, another inference method is to use approximate representations of distributions, which are typically mixtures of functions. Examples of such representations are *mixtures of truncated basis functions* [83] and *piecewise polynomials* [136]. Such approximation functions can be used to compute posterior probabilities, but no hard guarantees about the quality of the result can be given. The accuracy of the approximation is usually measured in terms of the *Kullback-Leibler divergence* [82] between the distributions, which does not directly relate to the error of the computed probabilities. This makes it hard to draw conclusions on the quality of probabilities and expectations derived from the results, which is essential for decision making.

Methods which put virtually no restrictions on the distributions used are based on *Markov chain Monte Carlo* (MCMC) sampling. Those methods can be very effective, but often have to be hand-tailored for specific problems. There are however frameworks providing MCMC inference for generic relational probabilistic languages [5, 114]. Recently significant progress has been made to improve upon known problems of MCMC methods, such as slow convergence for high-dimensional distributions [69] and distributions with near deterministic probabilities [88]. Despite those improvements, there is a qualitative difference between sampling methods and the method proposed in our work. No hard guarantees can be given about the error of a sampling estimation. Even under perfect circumstances, that is detailed knowledge about the approximated distributions is available, guarantees for approximations obtained by MCMC algorithms can only be given in terms of for instance the *Monte Carlo standard error*, with gives a probabilistic, but no hard guarantee. Even worse, in realistic cases, nothing is known about the distributions, so no guarantees can be given at all. The approximation might *pseudo convergence*, which means that is seems to have converged to a solution, which is actually not the case. Determining whether a *Markov chain* converged is only possible under complicated theoretical conditions [124]. In general, there is no known way to be sure a Markov chain actually converged (see e.g. [17, p. 21]). In contrast, PCLP pushes expressivity as far as possible, but still allows one to measure the quality, making it possible to decide whether more computation time or effort to acquire *evidence* should be invested to compute better results.

5.5.3 *Lifted Inference*

Another method to deal with intractability of probabilistic problems is lifted inference. The idea is that symmetries in inference problems, which are especially present in first-order models, can be exploited to significantly improve efficiency of inference. A survey of lifted inference approaches is provided by Kersting [80]. Lifted inference can also be applied to continuous distributions, which has been shown by Choi et al. [25]. PCLP inference may also profit from lifted inference, but this is beyond the scope of this thesis.

5.6 CONCLUSIONS

We showed that for PCLP we can perform exact inference with similar complexity as for the precise case, which is a quite unique property for an imprecise formalism. We developed a concrete inference algorithm, based on a generalisation of WMC. The algorithm is able to exploit the same kinds of structure as ordinary WMC, which is the state-of-the-art of precise probabilistic inference. This shows the benefit of our general semantic foundation integrating existing *artificial intelligence* (AI) approaches. The language can straightforwardly be extended with arbitrary constraint theories, such as ordinary CLP. Furthermore, for performing inference we can make use of existing algorithms from the field of LP and *constraint satisfaction*, in particular modern SMT solvers, and generalise probabilistic inference by WMC, inheriting the method's strengths.

6 EFFECTIVE APPROXIMATION OF HYBRID DISTRIBUTIONS WITH BOUNDED ERROR

In this chapter we propose an efficient approximation algorithm for computing *marginal* probabilities for hybrid problems, based upon the work presented in the previous chapters.

6.1 EXPLOITING LOGICAL STRUCTURE FOR FINDING EFFECTIVE APPROXIMATIONS

We have shown in Chapter 4 that one can approximate with bounded error a wide class of hybrid probabilistic *inference* problems in terms of *imprecise* distributions and in Chapter 5 that one can compute such approximations efficiently given a discretisation of the continuous distributions. What is missing to make such approximation practical is a way to obtain effective discretisations.

In this chapter we present a novel approximate inference algorithm, called the *iterative hybrid probabilistic model counting* (IHPMC) algorithm, yielding approximations with bounded error, by incrementally refining discretisations. Effective discretisations are found by exploiting the *logical* structure of problems, by adopting a *coarse-to-fine* approach, which means to exploit information from previous iterations to effectively decrease the error of approximations.

A comparison is made with sampling methods. Such methods are known to converge slowly when dealing with (near) deterministic probabilities and cannot handle observed rare events well. Some other approximate inference methods that are able to handle continuous distributions, e.g. [136], also suffer from rare observed events. All such methods share the drawback that they only provide weak guarantees about the quality of the estimates. In the worst case, one gets an approximation that seems to be near the exact result, but is completely wrong, a phenomenon that for sampling is known as *pseudo convergence*. In domains such as diagnosis and crisis management, where wrong decisions may have a huge impact, guarantees about the quality of approximations are desirable. The need to deal with rare events is also characteristic for such domains. We show that, in spite of the much stronger guarantees IHPMC provides on the quality of the result, our method is competitive and even superior for hybrid problems that possess a rich logical structure and concern observed rare events.

We first provide a motivating example to illustrate the importance of our work in Section 6.2. Then, in Section 6.3 we will define the probabilistic inference task considered. Subsequently, the novel IHPMC algorithm is introduced in Section 6.4. Finally, we experimentally evaluate the algorithm by comparing it with sampling-based methods (Section 6.5). Sections 6.6 and 6.7 discuss related work and conclude the chapter.

6.2 MOTIVATING EXAMPLE

As part of a model-based diagnostic system, we model the *event* of a failed component. Usually, the component is cooled down, but the cooling can also fail, modelled by the boolean *random variable* **NoC**. Suppose the environment temperature is modelled with continuous variable $\mathbf{T} \sim \mathcal{N}(20.0, 5.0)$. We can then express that the component can bear a higher temperature with cooling by modelling the event of failure with a *Probabilistic Constraint Logic Programming* (PCLP) program as:

$$\begin{aligned}&\mathit{fail} \leftarrow \langle \mathbf{T} > 20.0\rangle, \langle \mathbf{NoC} = \mathit{true}\rangle\\&\mathit{fail} \leftarrow \langle \mathbf{T} > 30.0\rangle\end{aligned}$$

The probability of failure can be computed exactly if it is assumed that the *cumulative distribution function* (CDF) of the Gaussian distribution can be computed exactly, thereby ignoring the potential error of typical implementations of such functions, as such errors are typically negligible and predictable.

As already discussed in Section 4.5.1.2, the situation is different in case we want to compute the probability of a constraint between a number of continuous random variables with potentially different kinds of distributions. For example, in case the temperature the component can bear is uncertain as well, we may want to compute the probability of $\mathbf{T} > \mathbf{L}$, where $\mathbf{L}$ represents the temperature limit of the component. In this case exact computation of the probability is impossible and one has to resort to approximating this probability. Such methods are either based on sampling or some approximate representation of CDFs that supports exact inference. If at all, only probabilistic guarantees for the quality of such approximations can be given. However, in case the probability of the *evidence* is very low, i.e. one observes a rare event, the quality of approximations decreases, and one may get a completely wrong result without realising it.

This is problematic in case one wants to base a decision on such probabilities. For instance, the decision whether or not to stop a machine and check a particular component may be based on the trade-off between the costs of stopping the machine, on the one hand, and machine failure, on the other hand with associated uncertainties. The work in this chapter provides a way to compute approximations with bounded, known error for such problems, which allows one to always find the best decision with certainty.

6.3 PROBABILISTIC INFERENCE TASK

As in Chaper 5, we will use a representation, in which events are represented by *propositional* logic formuals extended with constraints, as in *satisfiability modulo theories* (SMT) problems. Distributions are however precise, in contrast to the inference setting in Chaper 5. This can be seen as a general assembly language for hybrid probabilistic inference. How to convert PCLP programs to such representation was discussed in Section 5.1.

6.3.1 *Logical Representation Language*

The representation language is based on the key insight that any marginal probability can be mapped to a probability of an event defined on *independent* random variables only, expressed by a logical formula. Dependencies in the original distribution are reflected through the logical structure.

We use the following representation language as a basis. We consider a finite number of random variables $\mathbf{V}_1, \ldots, \mathbf{V}_n$ which have a *support*, i.e. the range of values on which the distribution is defined. The support of each random variable forms a measurable sub-space. These supports are in principle arbitrary; we in this chapter consider binary and real-valued random variables as typical representatives of discrete and continuous variables. The method can however be applied to a wide range of possible kinds of supports.

To each random variable $\mathbf{V}$ we associate an *event space* $\mathcal{A}_{\mathbf{V}}$. Such event spaces are the basis for the events that can be represented. For the boolean variables, the events space is obviously the power set of the supports and for the real variables we consider the set of all open and closed intervals with rational bounds.

Furthermore, we assume that all random variables are distributed independently and that we can compute the probability of all events. For the continuous variables, as discussed, we assume that their CDFs can be computed exactly, thereby ignoring the potential error of typical implementations of such functions, as such errors are typically negligible and predictable.

This together defines a *probability space* in the sense of Kolmogorov's *probability theory* [81]. The measure's *sample space* Ω is the product of all random variables' supports and its event space $\mathcal{A}$ the *tensor-product σ-algebra* of the random variables' event spaces. The *probability measure P* is defined by the independent probability measures.

To represent events in a structured way we use logical formulas with constraints on the variables. The basic building block of formulas are primitive constraints. For the boolean variables, the possible constraints are equality and its negation ($=,\neq$) between other boolean variables and the constants *true* and *false*. For the real-valued variables, we consider linear arithmetic constraints (e.g. $2\mathbf{X} \leq \mathbf{Y} + 2$). The method can be applied to arbitrary ranges and associated constraint theories, given that satisfiability of constraints is decidable.

Events in the combined probability space are represented by formulas, which are compositions of primitive constraints by arbitrary propositional logical connectives ($\wedge$, $\vee$, $\neg$, $\ldots$).

Example 6.1

Consider again the PCLP program from Section 6.2:

$$\begin{aligned}
&\mathit{fail} \leftarrow \langle \mathbf{T} > 20.0\rangle, \langle \mathbf{NoC} = \mathit{true}\rangle \\
&\mathit{fail} \leftarrow \langle \mathbf{T} > 30.0\rangle
\end{aligned}$$

The event that the component fails can be represented by the formula:

$$(\mathbf{NoC} = \mathit{true} \wedge \mathbf{T} > 20.0) \vee \mathbf{T} > 30.0$$

6.3.2 *Inference Problem*

We split the random variables into discrete ones $\mathbf{V}_1, \ldots, \mathbf{V}_l$ and continuous ones $\mathbf{V}_{l+1}, \ldots, \mathbf{V}_n$ and denote the spaces of all possible *valuations* as $\Omega_{\mathcal{D}}$, which is the product of all discrete random variables' supports, and $\Omega_{\mathcal{C}}$, which is the product of all continuous random variables's supports. The probability of any event e, represented as logical formula, is then defined as:

$$P(e) \overset{\text{def}}{=} \sum_{v_1,\ldots,v_l \in \Omega_{\mathcal{D}}} \underset{\Omega_{\mathcal{C}}}{\int \cdots \int} \mathbf{1}_e(v_1, \ldots, v_n) \prod_{i=1}^{l} P_i(v_i) \prod_{i=l+1}^{n} p_i(v_i)\, \mathrm{d}v_{l+1}, \ldots, v_n$$

Here $\mathbf{1}_e$ is an indicator function, indicating whether a valuation is an element of the event represented by e or not, P_i is the discrete probability measure of $\mathbf{V}_i$ and p_i is the *probability density function* (PDF) of continuous random variable $\mathbf{V}_i$.

The inference task we consider in the remainder is computing the probability of an event e given observations o:

$$P(e \mid o) = \frac{P(e \wedge o)}{P(o)},$$

where both e and o are expressed as logical formulas. Computing $P(e)$ and $P(e \mid o)$ is in general not possible, as it may require to compute infinite sums and uncomputable integrals.

6.4 ITERATIVE HYBRID PROBABILISTIC MODEL COUNTING

In this section we discuss the novel IHPMC algorithm.

6.4.1 *Hybrid Probability Trees*

Probability trees are a basic concept to compute marginal event probabilities [142] and were already employed in Chapter 5. In short, these trees represent a particular choice for the value of a particular random variable at every edge such that the paths through the tree constitutes the sample space. Such trees can be used to compute probabilities of an event by considering the paths of the tree where this event occurs. The idea is related to that of logic *semantic trees* [131], where a particular value of an *atom* is chosen at each edge and the formula is conditioned on the choices made, which can be repeated until the formula simplifies to $\bot$ or $\top$. In this chapter, we use a combination of both, where nodes contain logical formulas and where the children of a node represent a partitioning of *possible worlds*. Furthermore, these trees can be used to compute the probability of events, expressed by logical formulas.

The main idea is similar to the principles used in solvers for binary *weighted model counting* (WMC) [133]. However, whereas in previous work this is applied to discrete, finite ranges, we extend the concept to continuous random variables. In order to do so, at each node we split the variable's range into two parts and allow for a further split at a deeper level. Then, for each choice taken at a certain child, the probability is assigned that the random variable takes a value in the chosen partition, conditioned on the range chosen for the random variable earlier on the path. As discussed, such probabilities can be computed for continuous random variables by their CDFs, because of the initial independence of the random variables.

Definition 6.1 (Hybrid Probability Trees). *A* hybrid probability tree *(HPT) is a binary tree with at each node n a propositional formula φ_n and for all random variables $\mathbf{X}$ a range denoted by range$(n, \mathbf{X})$. If r is the root node, it holds that range$(r, \mathbf{X})$ equals the support of $\mathbf{X}$. Now let n be some node with children $\{c_1, c_2\}$. To each edge $n \rightarrow c_i$, $i \in \{1, 2\}$, a particular range τ_{ni} is associated to a fixed random variable $\mathbf{Y}$ such that $\tau_{n1} \cup \tau_{n2} = range(n, \mathbf{Y})$ and $\tau_{n1} \cap \tau_{n2} = \varnothing$. It holds that $range(c_i, \mathbf{Y}) = \tau_{ni}$ and $range(c_i, \mathbf{Z}) = range(n, \mathbf{Z})$ if $\mathbf{Y} \neq \mathbf{Z}$, with $i \in \{1, 2\}$. Furthermore, the formula φ_i associated to c_i equals φ_n simplified by the restrictions on the range on $\mathbf{Y}$ at c_i, i.e. by observing that some primitive constraints can be replaced by $\top$ or $\bot$. Furthermore, to each edge $n \rightarrow c_i$, with $i \in \{1, 2\}$, a probability p_{ni} is assigned, such that $p_{ni} = P(\mathbf{Y}_n \in \tau_{ni} \mid \mathbf{Y}_n \in range(n, \mathbf{Y}))$. Finally, if l is a leaf node, then it holds that $\varphi_l = \top$ or $\varphi_l = \bot$.*

Given a particular HPT, it is straightforward to compute the probability of the event represented by the formula at the root node. First, the probability of a leaf is by definition 0 or 1. Furthermore, due to the properties of the tree it is easy to show that for each non-leaf node n the probability of φ_n, given the ranges of random variables, can be computed by $p_n = p_{n1} \cdot p_1 + p_{n2} \cdot p_2$, where p_1 and p_2 are the probabilities associated to n's children. So the probabilities can be computed from the bottom to the top of the tree.

Example 6.2

Consider again the event that a component fails, as represented in Example 6.1:

$$(\mathbf{NoC} = \mathit{true} \land \mathbf{T} > 20.0) \lor \mathbf{T} > 30.0$$

Suppose continuous variable $\mathbf{T} \sim \mathcal{N}(20.0, 5.0)$ is used to model the temperature. Whether the cooling fails is modelled by the boolean random variable $\mathbf{NoC} \sim \{0.01\colon \mathit{true}, 0.99\colon \mathit{false}\}$.

A possible HPT is given in Figure 6.1. At each node we simplify the formula as much as possible given the choices made. The probabilities for the continuous variables are rounded to four decimals. Note that the probability assigned to the edge with choice $\mathbf{T} \in (-\infty, 20]$ is not $P(\mathbf{T} \in (-\infty, 20]) = 0.5$, as the range of $\mathbf{T}$ is already restricted on the path before. Therefore, it is $P(\mathbf{T} \in (-\infty, 20] \mid \mathbf{T} \in (-\infty, 30]) \approx 0.5116$. From this tree, the probability of component failure can be computed by $0.9772 \cdot 0.01 \cdot 0.4884 + 0.0228 \approx 0.0276$.

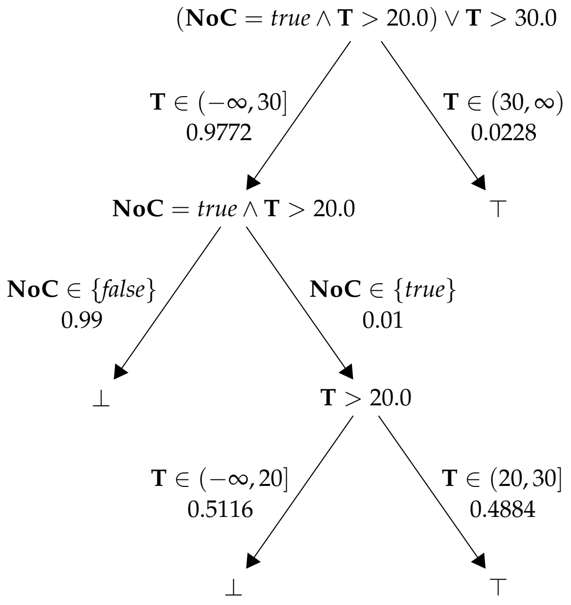


Figure 6.1: Example HPT

6.4.2 *Partially Evaluated Hybrid Probability Trees*

The concept of HPTs is restricted to problems for which exact inference is possible. For continuous events, which cannot be represented by *hyperrectangles*, we can never simplify all leaves to $\perp$ or $\top$. We tackle this problem by using *partially evaluated hybrid probability trees* (PHPTs). The only difference between HPTs and PHPTs is that for PHPTs we drop the requirement that all leaves must contain

$\perp$ or $\top$ and allow arbitrary formulas. One can therefore further extend a PHPT either ad infinitum or until one finds a HPT. In the next section we show how PHPTs can be used to compute approximations with known maximal error.

Example 6.3

Suppose the limit of the temperature the component can bear is uncertain as well, modelled by $\mathbf{L} \sim \mathcal{N}(30.0, 5.0)$. A PHPT for the event that the temperature is above the limit is depicted in Figure 6.2. It is impossible to define a finite HPT for this problem, as the event does not have the shape of a hyperrectangle. Note that the formula $\mathbf{T} > \mathbf{L}$ often cannot be simplified directly, so the information about all chosen ranges upwards on the path is required. This is in contrast to binary versions of model counting algorithms, for which the choices of variable values are completely reflected in the simplified formula.

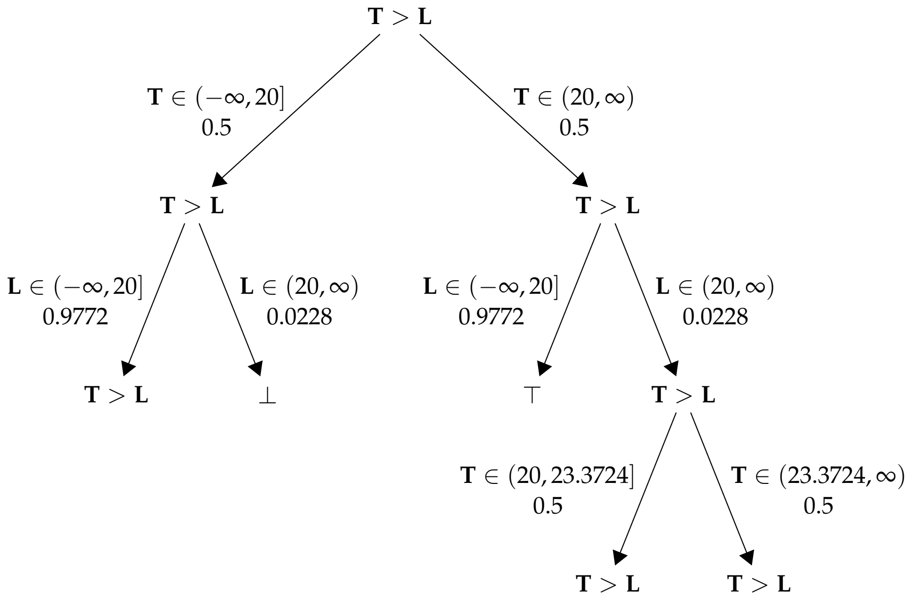


Figure 6.2: Example PHPT

6.4.3 *Approximating Probabilities by Partially Evaluated Hybrid Probability Trees*

We here show how probabilities are approximated by employing PHPTs.

6.4.3.1 *Bounds on Event Probabilities*

Each leaf of a PHPT with a formula which is not $\perp$ or $\top$ corresponds to an area of the sample space which is only partially part of the event associated to the root node. While this does not provide an exact probability of this event, it does provide a bound on this probability by assuming the probability of such areas is 0.0 or 1.0 respectively. In this way, PHPTs can be used to compute probability bounds, denoted by $\underline{P}$ and $\overline{P}$.

Example 6.4

The PHPT in Figure 6.2 contains only one leaf with $\top$, whereas all other leaves contribute at most partially to the event's probability. The lower bound of the event's probability is therefore the probability associated to this single leaf: $\underline{P}(\mathbf{T} > \mathbf{L}) = 0.5 \cdot 0.9772 = 0.4886$. Analogously, as there is only one path which certainly does not contribute to the probability, the probability's upper bound is $\overline{P}(\mathbf{T} > \mathbf{L}) = 1 - 0.5 \cdot 0.0228 = 0.9886$.

Lemma 6.1. *For any event e and PHPT for e:*

$$\underline{P}(e) \leq P(e) \leq \overline{P}(e)$$

Furthermore, we can always achieve arbitrary precisions, by evaluating the PHPT sufficiently deep.

Lemma 6.2. *For every event e and every maximal error ϵ, there is a PHPT, such that:*

$$P(e) - \underline{P}(e) \leq \epsilon \wedge \overline{P}(e) - P(e) \leq \epsilon$$

6.4.3.2 *Bounds on Conditional Event Probabilities*

Conditional probabilities are defined as $P(e \mid o) = P(e \wedge o)/P(o)$. As this definition requires the non-conditional probabilities of two events, we cannot compute bounds on this probability using a single PHPT, but have to use a second one. We can furthermore compute tighter bounds by making use of the fact that the bounds of $e \wedge o$ and e are not independent: an area cannot at the same time be outside of e, but within $e \wedge o$. This can be exploited by expressing the bounds of conditional probabilities in terms of bounds on two non-conditional probabilities, as we have shown in Chapter 4 (Proposition 4.3):

$$\underline{P}(q \mid e) = \frac{\underline{P}(q \wedge e)}{\underline{P}(q \wedge e) + \overline{P}(\neg q \wedge e)}$$
$$\overline{P}(q \mid e) = \frac{\overline{P}(q \wedge e)}{\overline{P}(q \wedge e) + \underline{P}(\neg q \wedge e)}$$

Also for conditional probabilities we can therefore achieve arbitrary precisions.

Proposition 6.1. *For all events e, o and maximal error ϵ, there are PHPTs, such that:*

$$P(e \mid o) - \underline{P}(e \mid o) \leq \epsilon \wedge \overline{P}(e \mid o) - P(e \mid o) \leq \epsilon$$

6.4.4 *Anytime Inference by Iterative Hybrid Probability Tree Evaluation*

The anytime approximation IHPMC algorithm based on PHPTs is given in Algorithm 3. We start with two PHPTs with initial root nodes $e \wedge o$ and $\neg e \wedge o$, which we use to compute in each iteration an approximation according to Proposition 4.3. In case no evidence is present, we can use a single PHPT with formula e. In the general case, at each iteration we first choose one of the two PHPTs and then, in this PHPT, we choose a non-evaluated node to evaluate further. From the formula of this node we choose a random variable that will be used for splitting and finally we choose a partitioning for this random variable.

Algorithm 3: IHPMC Algorithm

```
   Input: Events $e$ and $o$, maximal error $\epsilon$
   Result: Approximation of $P(e \mid o)$ with maximal error $\epsilon$
 1 $phpt\_pos$ = PHPT with single root node $e \wedge o$
 2 $phpt\_neg$ = PHPT with single root node $\neg e \wedge o$
 3 $p\_min = 0.0$
 4 $p\_max = 1.0$
 5 while $(p\_max - p\_min)/2 > \epsilon$ do
 6     $phpt\_to\_eval = choose\_phpt(phpt\_pos, phpt\_neg)$
 7     $node\_to\_eval = choose\_node(phpt\_to\_eval)$
 8     $rvar\_to\_branch = choose\_rvar(node\_to\_eval)$
 9     $(rvar\_part\_l, rvar\_part\_r) = choose\_part(rvar\_to\_branch, node\_to\_eval)$
10     add children to $node\_to\_eval$ according partition $(rvar\_part\_l, rvar\_part\_r)$
11     $p\_min = lower(phpt\_pos) / (lower(phpt\_pos) + upper(phpt\_neg))$
12     $p\_max = upper(phpt\_pos) / (upper(phpt\_pos) + lower(phpt\_neg))$
13 return $(p\_min + p\_max)/2$
```

Note that we could also in the algorithm compute components of the formula independently in case they do not share any variables, as usually done in WMC solvers, which we however leave out here for brevity. While the basic algorithm is straightforward, there are four non-deterministic choices at each iteration which largely determine the effectiveness of the algorithm. Of course, depending on the problem at hand, there will be different choices of heuristics that will be useful. In the following, we describe general heuristics which seem to be very effective for most problems.

As the first choice, we pick the PHPT which has the highest difference between lower and upper bound, i.e., the largest error. Second, we choose the non-evaluated node which represents the sub-space with the highest probability mass, as these nodes have the most potential for reducing the error. A more sophisticated heuristic, that tries to predict how much effort is required to reduce the error of the node's formula, offered no improvement in experiments. The reason is that an accurate prediction takes as much time as just trying to reduce the error and continuing with another branch if the error for the firstly chosen one cannot be reduced quickly.

In contrast to the choice of the node, the heuristic for choosing a random variable must be a good predictor for how much this simplified the formula, as this choice influences all descendants of the node in the tree. The issue is similar to choosing a variable in binary model counting, but the concept has to be generalised for continuous variables. We use a heuristics based on the observation that variables occurring more frequently simplify larger parts of the formula. This heuristic is similar to the *dynamic largest combined sum* heuristic, commonly used by binary model counters [134, Section 3.2] and used for the GWMC algorithm (Section 5.2.2.1). For continuous variables however we have to adapt the heuristic to make sure that random variables are not selected repeatedly without an opportunity to eliminate a primitive constraint, and thereby simplifying the formula.

Finally, we need to pick a partitioning of the variable. For the binary variables there is obviously only a single partitioning possible ($\{\mathit{true}\}$ and $\{\mathit{false}\}$). The partitioning of a continuous random variable amounts to picking a point $x \in \mathrm{range}(n, \mathbf{Y})$ of the range of the random variable $\mathbf{Y}$ at node n, which provides two sub-intervals $(-\infty, x] \cap \mathrm{range}(n, \mathbf{Y})$ and $(x, \infty) \cap \mathrm{range}(n, \mathbf{Y})$. Initially, we find points that can lead to a simplified sub-formula. For instance, in Figure 6.2 the choice of the point 20 in the left branch, leading to the restriction $\mathbf{L} \in (20, \infty)$, can be used to simplify the formula to $\perp$, given the choice for $\mathbf{T}$ above in the tree. In case there are multiple of those points, we count how many sub-formulas can be simplified by choosing this point. There are also cases in which there is no such point, as at the top or the rightmost evaluated node in Figure 6.2. In this case we use a heuristic trying to maximise the probability of the path eliminating the constraint.

Theorem 6.1. *IHPMC (Algorithm 3) terminates, i.e. it always finds an approximation with the desired error bound in finite time, given the heuristics as described above.*

6.5 EXPERIMENTS

We implemented the algorithm in the functional programming language *Haskell*. The implementation is available at `http://www.steffen-michels.de/ihpmc`. We compare IHPMC to sampling approaches, as sampling is the commonly used inference method for hybrid distributions. We compare to the sampler implementations of *BLOG* 0.8[1] [102] and *distributional clauses* (DC)[2] [61]. Both are optimised for problems with logical structure. BLOG provides several sampler implementations. We compare to the Likelihood weighted (BLOG LW) and the naive rejection sampler (BLOG RJ). The *Markov chain Monte Carlo* (MCMC) sampler of BLOG fails completely on the problem presented. To allow a comparison, we first ignore the bounded error guarantee of IHPMC and just consider the

1 `http://bayesianlogic.github.io`
2 `https://code.google.com/p/distributional-clauses`

error of approximations. We measure the squared error and for the sampling approaches the mean squared error over 50 runs versus inference time. All experiments were run on a laptop with a *Intel Core i3* 2.4 GHz processor and 4GB of RAM.

6.5.1 *Benchmark Problem*

We use an abstract diagnostic problem, which is a typical example of a domain were events (failures) are rare. Also such problems possess a rich logical structure and are of hybrid nature. Our model includes three kinds of reasons why a component can fail: discrete reasons occurring with some probability, continuous reasons meaning that a continuous variables exceeds some limit which is uncertain itself (e.g. the temperature got too high), and another failing component. In our abstract model the failure of a component $i > 0$ is modelled as:

$$fails_i \leftrightarrow \mathbf{DCause}_i = true \vee \mathbf{CCause} > \mathbf{Limit}_i \vee fails_{i-1}$$

For $i = 0$, the component does not depend on other components, so it can only fail because of the first two reasons. The number of components n and probability p of $\mathbf{DCause}_i$ are varied in the experiments. For the continuous variables we use the following distributions: $\mathbf{CCause} \sim \mathcal{N}(20.0, 5.0)$ and $\mathbf{Limit}_i \sim \mathcal{N}(\mu, 5.0)$, where μ is a parameter we also vary during the experiments. Note that since we use the same continuous cause variable for all components, dependencies are created such that probabilities cannot be computed exactly in a straightforward way. Also note that variables are all normally distributed for simplicity. The IHPMC algorithm can also deal with mixtures of different distributions, which can be achieved by just using different CDFs.

6.5.2 *Results*

Here we discuss the results of exploring the performance of the algorithms with varying parameters.

6.5.2.1 *Non-Conditional Probabilities With Varying Parameters*

The results depicted in Figure 6.3 are based on a run with parameters where the continuous cause has a high probability. It is much higher than for a realistic diagnostic problem, but is used to illustrate behaviour of the algorithm. In the first milliseconds the error of IHPMC is very unstable, as the bound on the error is still relatively large. Actually, it takes about 2.6 seconds before the error is guaranteed to be below 0.01. Decreasing the error bound can temporally result in an increased error. As the error of the sampling algorithms are averaged over several runs, their error decreases more or less monotonically. IHPMC can profit from the case in which the discrete cause has more impact, as shown by the experiment in Figure 6.4. The error bound drops below 0.01 already after about

6 milliseconds. The sampling algorithms can also profit from the discrete structure, but are clearly outperformed by IHPMC. Also, IHPMC can profit when the continuous part of the event is more rare, which is more realistic for a diagnostic problem (Figure 6.5). Here is takes about 28 milliseconds to get an error bound of 0.01. In this case, IHPMC cannot make use that much of the discrete structure, but in the first iterations a single path with a high probability is found that implies $\mathbf{DCause}_0 = \mathit{false} \wedge \mathbf{CCause} \leq \mathbf{Limit}_0 \wedge \cdots \wedge \mathbf{DCause}_{n-1} = \mathit{false} \wedge \mathbf{CCause} \leq \mathbf{Limit}_{n-1}$. This disproves fails_{n-1}, which means the upper probability bound jumps to a small value.

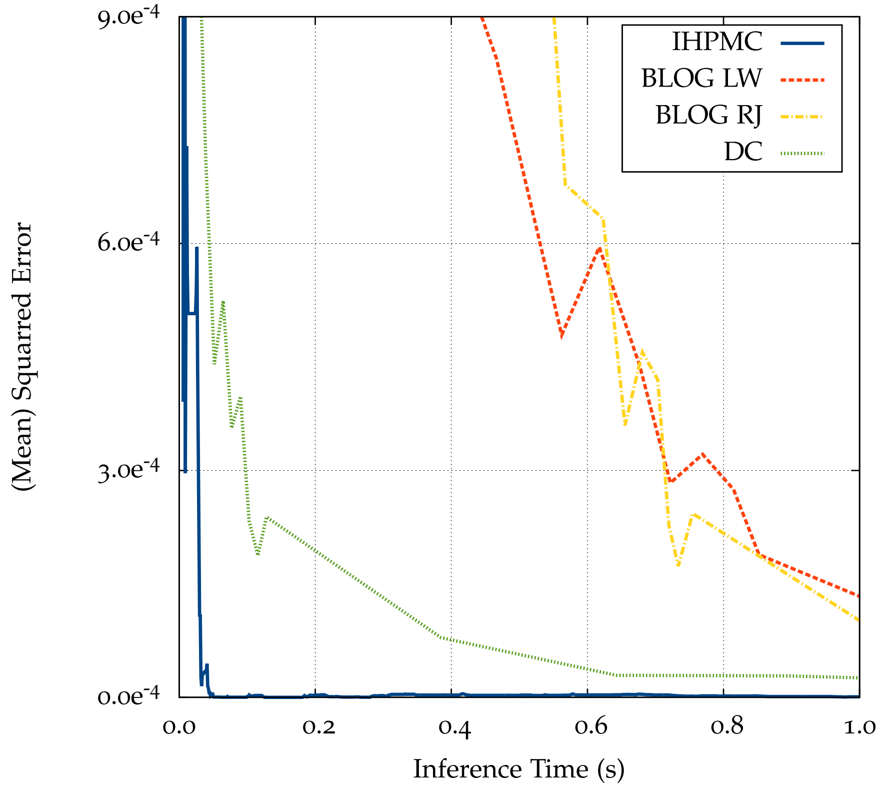


Figure 6.3: Approximation Errors ($P(\mathit{fails}_9)$, $n = 10$, $p = 0.01$, $\mu = 30.0$)

6.5.2.2 *Scalability*

To show scalability of IHPMC, we performed an experiment with a larger set of components ($n = 100$), where it takes about 2.3 seconds to get a guaranteed error below 0.01. As the results in Figure 6.6 show, in the beginning two of the

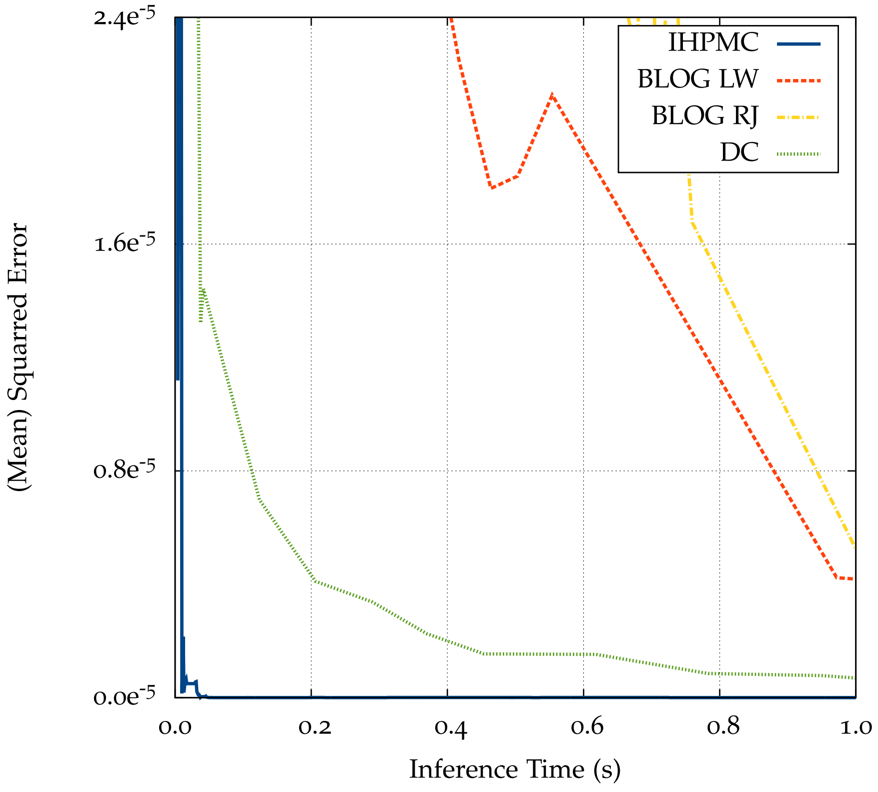


Figure 6.4: Approximation Errors ($P(\mathit{fails}_9)$, $n = 10$, $p = 0.3$, $\mu = 30.0$)

sampling implementations outperform IHPMC, as sampling algorithms by their nature do not suffer from a high number of dimensions. However, recall that IHPMC is outperformed by algorithms providing much less information, as no guarantees about the error are provided by the sampling algorithms. After about 1.5 second however the results of IHPMC become competitive.

6.5.2.3 *Computing Probabilities Conditioned on Rare-Event*

Finally, in Figure 6.7, we show that IHPMC is superior if probabilities conditioned on rare events are computed. The inference task is quite natural for a diagnostic setting: the failure of some component is observed and one wants to know whether the failure is caused by some subcomponent. The performance of IHPMC is also affected by conditioning on the rare event; it takes about 250 milliseconds to guarantee an error below 0.01, while for the probability of the

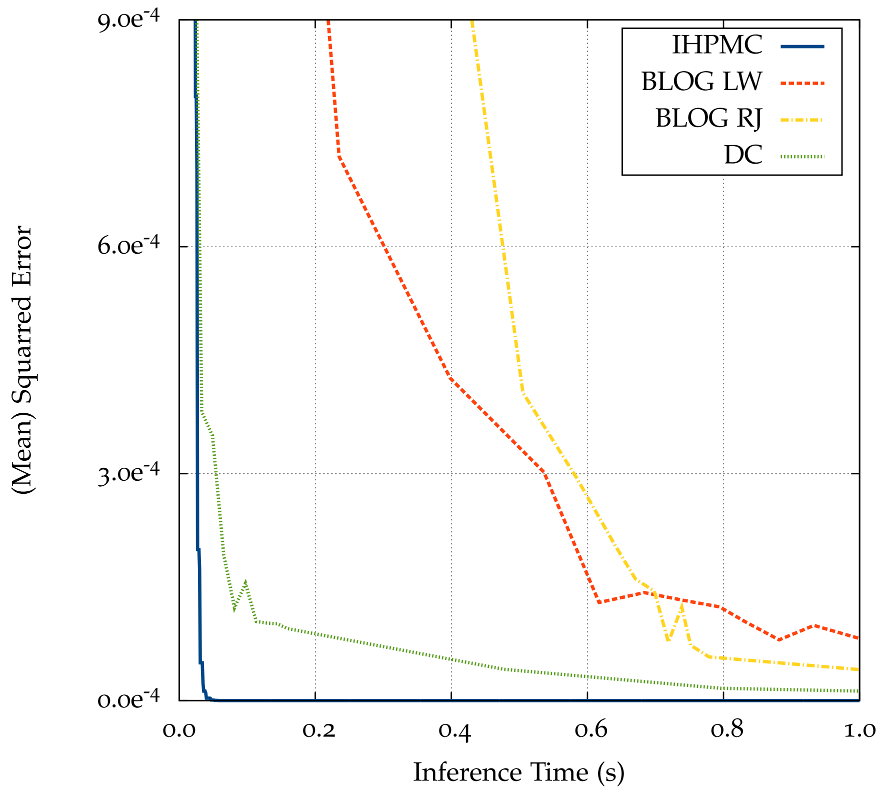


Figure 6.5: Approximation Errors ($P(\mathit{fails}_9)$, $n = 10$, $p = 0.01$, $\mu = 60.0$)

evidence only this takes about 15 milliseconds. The performance of the samplers however decreases significantly more.

Error measures only tell half of the story as IHPMC provides a different kind of result as sampling methods. This is illustrated in Figure 6.8, where the bounds computed by IHPMC have been visualised together with point approximations produced by a sampler. Each point represents a single run of the sampler, which can or cannot be within the bounds computed by IHPMC. However, as a user of a sampling method one only obtains a single point and there is no way to tell whether this is a good approximation or not.

6.6 RELATED WORK

Recently, an exact inference algorithm which exploits the logical structure of problems by employing constraint solvers, in this case for bounded integers, has

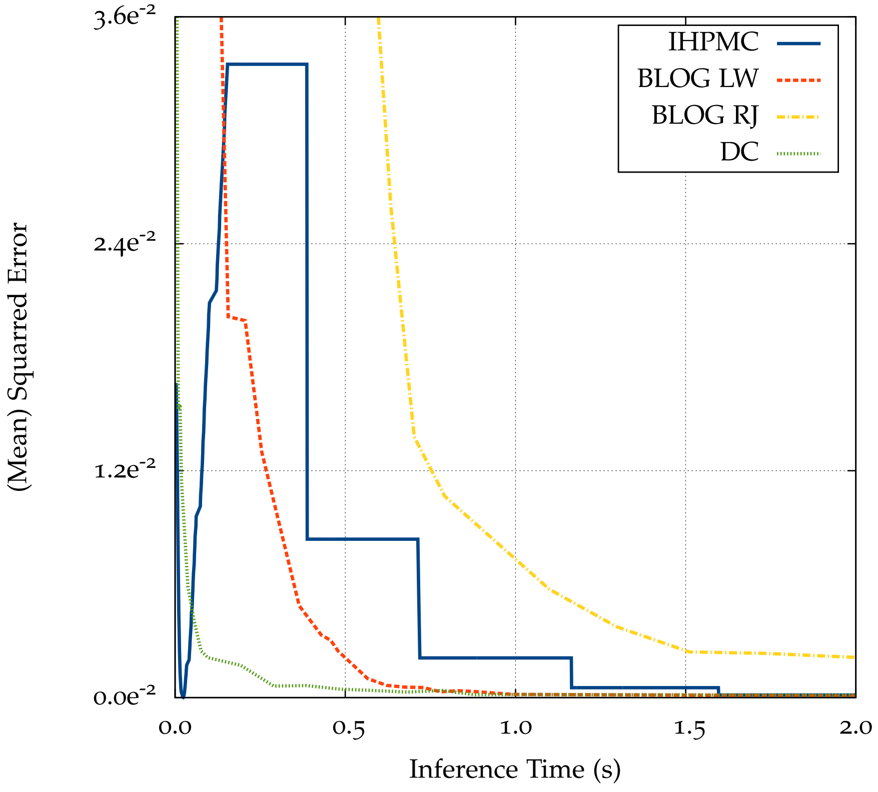


Figure 6.6: Approximation Errors ($P(fails_{99})$, $n = 100$, $p = 0.01$, $\mu = 60.0$)

been proposed [41]. Compared to the work presented in this chapter, this method does not work for unbounded integers and continuous distributions, which is a major restriction.

There are several approximation methods for hybrid problems that use approximate representations for continuous distributions for which exact inference is possible. Examples of such representations are *mixtures of truncated basis functions* [83] and *piecewise polynomials* [136]. Recently such ideas have also been combined with ideas from logical WMC, similar to the work presented in this chapter [8]. These methods have in common that they do not provide guarantees about the error of computed posterior probabilities with respect to the original distribution. It may seem that approximate representations, such as piecewise polynomials, can resemble intended distributions very closely, but conditioning on evidence with small probability can significantly increase the error made when computing marginal probabilities.

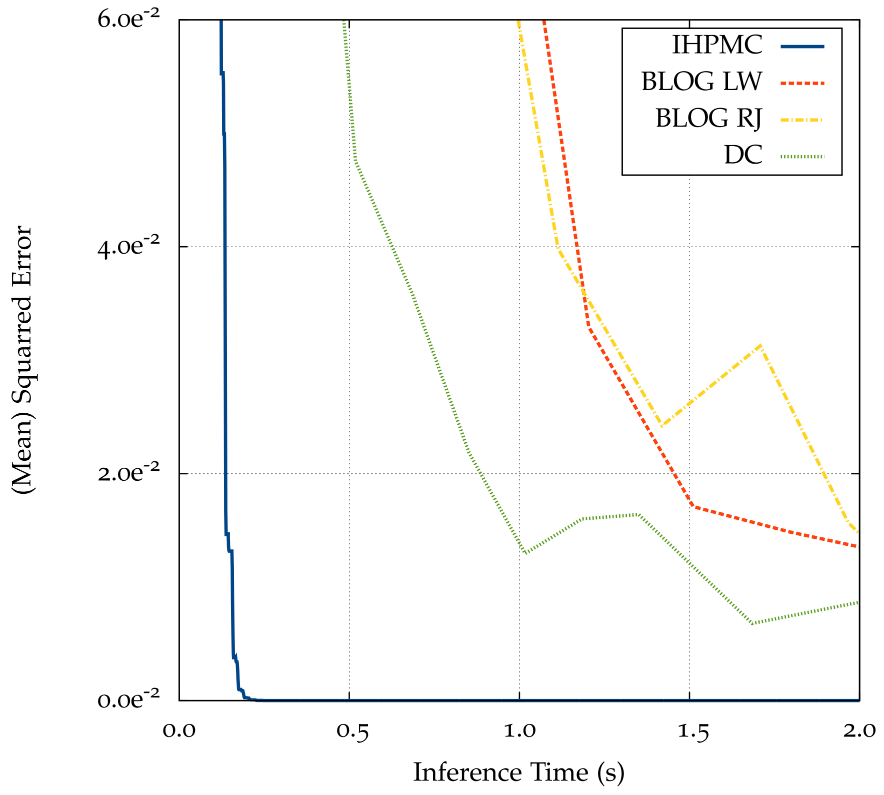


Figure 6.7: Approximation Errors ($P(fails_9 \mid fails_0)$, $n = 10$, $p = 0.0001$, $\mu = 60.0$)

Another class of approximation methods, virtually putting no restrictions on the type of probability distribution and evidence, are based on sampling. While the performance of straightforward sampling methods decreases dramatically when conditioning on rare events, MCMC methods can be very effective if they are hand tailored for specific problems. However, generic MCMC frameworks are usually suboptimal [5, 114], while it is desirable to have a general tool to perform inference on various models, as IHPMC does. Despite of many improvements solving many issues with sampling methods, such as slow convergence for high-dimensional distributions [69] and distributions with near deterministic probabilities [88], there is a qualitative difference between sampling methods and the method proposed in this chapter: even under perfect circumstances, when detailed knowledge about the approximated distributions is available, guarantees for approximations obtained by MCMC algorithms can only be given in terms of for instance the *Monte Carlo standard error*, which gives a probabilistic, but no

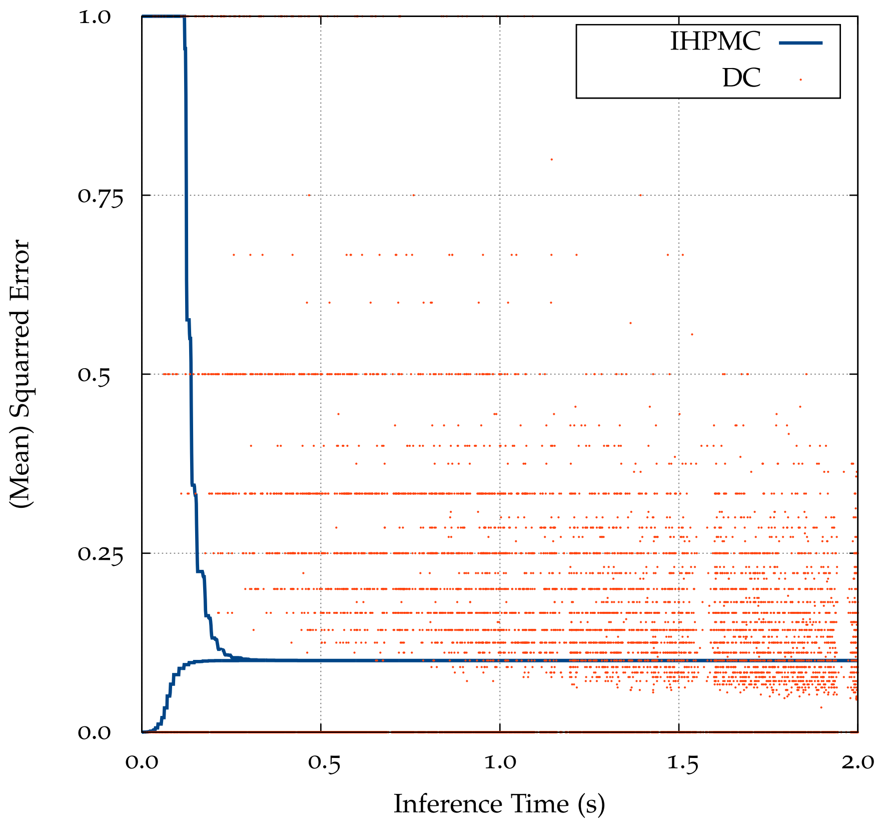


Figure 6.8: Probability Approximations ($P(\mathit{fails}_9 \mid \mathit{fails}_0)$, $n = 10$, $p = 0.0001$, $\mu = 60.0$)

hard guarantee, as the work in this chapter provides. Even worse, in realistic cases, nothing is known about the distributions, so no guarantees can be given at all. The approximation might pseudo converge, which means that is seems to have converged to a solution, which is actually not the case. Determining whether a *Markov chain* converged is only possible under complicated theoretical conditions [124]. In general, there is no known way to be sure a Markov chain actually converged (see e.g. [17, p. 21]).

Recent hashing-based sampling methods, which exploit logical structure of problems, have been extended for hybrid distributions [9, 24] as well. Such methods provide a guarantee on the quality of approximations, but this remains a probabilistic guarantee on non-conditional probabilities. In contrast, the algorithm presented in this chapter, provides a hard guarantee for computed conditional probabilities. Moreover, the methods mentioned above are restricted to exactly computable representation of continuous distributions (piecewise polynomials [9] and uniform distributions [24]), while we provide inference for arbitrary PDFs with computable CDFs.

Several approximation methods have been proposed that provide hard guarantees for discrete distributions [121]. Recently, the effectiveness of these kind of methods in combination with state-of-the-art knowledge compilation techniques has been investigated [128, 129, 153]. In contrast to IHPMC, these algorithms are restricted to discrete problems and do not handle evidence, which are significant limitations in practice.

6.7 CONCLUSIONS

In this chapter, we have introduced the first practical hybrid inference algorithm which provides a hard bound on the error of approximations of marginal probabilities with evidence. An important feature of the algorithm is that it exploits the logical structure of the problem, which makes it suitable for both propositional and first-order formalisms, such as logic-based formalisms for *statistical relational learning* (SRL) [54].

The proposed algorithm is applicable to a wide range of probabilistic inference problems provided that it can be decided whether primitive constraints on random variables are true or not given restrictions on the ranges of random variables, which is needed for building a PHPT. There are obviously theoretical limitations on this decision procedure; however, it is straightforward if the range-restrictions are modelled as intervals and events are modelled using linear constraints as illustrated in the examples.

For very high dimensional continuous problems with many dependences and little structure, the IHPMC algorithm may perform worse than sampling methods, which are not sensitive to high dimensional dependences. Many realistic problems stemming from SRL are however characterised by a moderate number of dimensions with a significant amount of structure. For such problems IHPMC is competitive with state-of-the-art sampling algorithms, in spite of the fact that it provides much stronger guarantees. It outperforms sampling methods by far if rare events with deterministic structure are provided as evidence. Moreover, even for problems for which our algorithm converges slowly, this is visible for the user and can therefore prevent wrong conclusions, which is in general not the case for MCMC methods.

7 IMPRECISE PROBABILISTIC HORN CLAUSE LOGIC

With the work on *Probabilistic Constraint Logic Programming* (PCLP), we have shown that one can employ *imprecise* reasoning with little additional cost in terms of *inference* efficiency. However, PCLP focuses on *independent* imprecise *random variable definitions*, while its expressivity is limited for conditional random variable definitions, e.g. compared to *locally defined credal networks* (LDCNs). In this chapter we introduce *imprecise probabilistic Horn clause logic* (IPHL), which aims towards imprecise knowledge representation involving conditional definitions, but as well avoids to increase inference complexity.

After explaining the basic properties of IPHL (Section 7.1), we first provide a motivating example demonstrating knowledge representation capabilities and limitations of other approaches in Section 7.2. Then the language is defined formally (Section 7.3) and the inference method is presented (Section 7.4). Finally, related work is discussed in Section 7.5 and Section 7.6 concludes the chapter.

7.1 IMPRECISE KNOWLEDGE-REPRESENTATION WITHOUT ADDITIONAL COST

As discussed before, there is in reality often not only uncertainty about what is true in the world, but also uncertainty about the exact probability distribution of how the world behaves. This hinders employing languages defining a precise probability distribution, as forcing to specify precise probabilities, not justified by the knowledge, can lead to conclusions that might be *sound* given the model, but not sound given the knowledge available. Such ignorance about how the world behaves can be expressed qualitatively with non-probabilistic *logics*, such as *first-order logic* (FOL), which allows one to leave the truth of statements open. *Alethic modal logics* provides more expressive means to represent ignorance by so-called *modalities*. Such modality can for instance express that a statement is *possibly true*. Modal logics allow dealing with uncertainty in a more powerful, but still merely qualitative way. As discussed, imprecise *probability theory* (Section 2.2.3) and in particular *probabilistic logics* defining *credal sets* (Section 3.2) allow one to express ignorance about probability distributions, but still make it possible to draw certain quantitative conclusions. Although PCLP can deal with imprecise probabilities, it is not as expressive as most other imprecise formalisms, as it does not support conditional definitions and is restricted to closed probability intervals. In particular, it cannot express alethic modal concepts as a special case.

In this chapter we propose a new language, IPHL, that integrates qualitative and quantitative uncertainty knowledge representation and reasoning. The semantics of IPHL is based on imprecise probability intervals, of which the bounds can be open or closed. This allows one to exactly define the semantics of qualitative modalities and also provides precise probabilistic logic as a special case. There are however restrictions in the structure of theories, as the language is designed with particular guarantees for computational tractability in mind. Inference complexity for IPHL is even lower than inference complexity for PCLP. Specifically, inference for IPHL has the same complexity as corresponding precise probabilistic inference problems, without the cost of an increased *treewidth* as for PCLP. IPHL is the first imprecise probabilistic language that offers such guarantees. We further propose a concrete inference mechanism based on the translation to an ordinary *weighted model counting* (WMC) problem. This confirms the lower inference complexity compared to PCLP, that requires the generalised GWMC algorithm with increased parametrised complexity.

7.2 MOTIVATING EXAMPLE

We first show knowledge representation capabilities and limitations of other logical languages by means of a motivating example.

7.2.1 *Logic Programming*

We start with an attempt to model the problem of the motivating example in *logic programming* (LP) (Section 2.1.3). In this chapter we restrict all rules to being *Horn clauses*, which means that there is no negation, so we can assume the traditional least model semantics of LP.

Example 7.1

Consider the following problem from the maritime safety and security domain. Suppose we want to model in which cases a vessel poses an environmental hazard and may for instance not enter certain restricted areas. One reason for a vessel being an environmental hazard is that it has some chemical substances loaded. We could model this using LP as:

$$env\_hazard \leftarrow chemicals$$

The problem is that the model actually expresses, that in case we know that the vessel has chemicals loaded, it certainly is an environmental hazard and otherwise it is certainly not, assuming the *closed world assumption* of LP. Clearly, there are vessels with chemicals, which are no environmental hazard, for instance because the amount is not significant, and ships without chemicals on board, which still are an environmental hazard.

7.2.2 *Modal Logic*

The idea of modal logics is to lift the restrictions that *propositional* statements are certainly true or false, by including operators that express modalities. Several practical implementations of programming languages based on modal logics are available [115]. There are different ways to define and interpret those modalities, but we restrict to classical alethic modalities, which express that something is possibly ($\Diamond$) or necessarily true ($\Box$).

Example 7.2

Example 7.1 can be made more precise using modal operators. For example, we could model the fact that having chemicals loaded makes it possible that the vessel is an environmental hazard with:

$$\Diamond \mathit{env\_hazard} \leftarrow \mathit{chemicals}$$

However, assuming the closed world assumption of LP, this rule alone implies that if the vessel has no chemicals on board, then it is not an environmental hazard. While we could try to sum up all the reasons for a vessel being an environmental hazard, it is a reasonable assumption that in reality we can never observe or even know all of those reasons. A possible solution is to state that a vessel always possibly poses an environmental hazard:

$$\Diamond \mathit{env\_hazard} \leftarrow \top,$$

where $\top$ denotes *true*. This is not useful in practice, since there are no cases for which we can draw a qualitatively different conclusion, which would be that the vessel certainly is a hazard. The rules above imply that *env_hazard* is possible whether or not the vessel is carrying chemicals. The knowledge that chemicals are a risk factor, increasing the likelihood of environmental hazard, cannot be expressed in the language.

Generally, modal logics can only be used to represent and reason about qualitative uncertainty, whereas the available quantitative knowledge cannot be used.

7.2.3 *Probabilistic Logic Programming*

We use a special probabilistic LP language for illustration, with support for probabilistic rules. This language serves as a basis for the work described in this chapter. The semantics of this language is a variant of Sato's *distribution semantics* (Section 3.4.2), extended with probabilities on rules instead of facts only.

Example 7.3

To include degrees of uncertainty, we extend Example 7.2 with probabilities as follows:

$$0.1\colon \mathit{env\_hazard} \leftarrow \top$$
$$0.4\colon \mathit{env\_hazard} \leftarrow \mathit{chemicals}$$

This states that, if a vessel has chemicals loaded, it will cause an environmental hazard with a probability of 0.4. The first rule represents other causes we do not model explicitly with a low probability.

We formally introduce the language. A theory consists of a set of probability-rule pairs $\mathbf{R}$:

$$\begin{aligned} &p_1\colon (h_1 \leftarrow b_{11}, \ldots, b_{1m_1}) \\ &\ldots \\ &p_n\colon (h_n \leftarrow b_{n1}, \ldots, b_{nm_n}) \end{aligned}$$

Then to each subset $\mathbf{S} \subseteq \mathbf{R}$, a probability is assigned:

$$P_{\mathbf{S}} \overset{\text{def}}{=} \prod_{p\colon (h \leftarrow b_1,\ldots,b_n) \in \mathbf{S}} p \prod_{p\colon (h \leftarrow b_1,\ldots,b_n) \in \mathbf{R} \setminus \mathbf{S}} 1 - p \tag{7.1}$$

In this chapter, we assume that there is a finite number of ground terms, which means that we can look upon each program as a propositional one by replacing all variables by all ground terms. As a result, there will be a finite number of subsets; however, the approach can be generalised to the full first-order case with an infinite number of ground terms, as shown in the original distribution semantics [137] and the work on PCLP in this thesis (Chapter 4).

For each such subset we can determine whether a query q holds ($\mathbf{S} \models q$) under the least model semantics of LP. Then, the probability of a query q given the rules of a program $\mathbf{R}$, is defined as the sum of the probabilities of all rule subsets for which the query can be derived:

$$P_{\mathbf{R}}(q) \overset{\text{def}}{=} \sum_{\substack{\mathbf{S} \models q \\ \mathbf{S} \subseteq \mathbf{R}}} P_{\mathbf{S}} \tag{7.2}$$

Example 7.4

The query *env_hazard* can be derived for the following subsets of rules in Example 7.3 assuming *chemicals* is true:

$$\begin{aligned} \mathbf{S}_1 &= \{\ 0.1\colon \mathit{env\_hazard} \leftarrow \top\ \} \\ \mathbf{S}_2 &= \{\ 0.4\colon \mathit{env\_hazard} \leftarrow \mathit{chemicals}\ \} \\ \mathbf{S}_3 &= \{\ 0.1\colon \mathit{env\_hazard} \leftarrow \top, \\ &\qquad 0.4\colon \mathit{env\_hazard} \leftarrow \mathit{chemicals}\ \} \end{aligned}$$

We have the following probabilities: $P_{\mathbf{S}_1} = 0.1 \cdot (1 - 0.4) = 0.06$, $P_{\mathbf{S}_2} = 0.36$ and $P_{\mathbf{S}_3} = 0.04$. Therefore $P_{\mathbf{R}}(\mathit{env\_hazard}) = 0.06 + 0.36 + 0.04 = 0.46$. This actually corresponds to the *noisy-or* combination of both rules' probabilities.

A limitation of such probabilistic approaches is that they require the precise quantification of likelihoods. This is often infeasible, for instance in domains dealing with very rare *events*, which are not observed often. For example, estimations of the probability that a vessel without chemicals on board is an environmental hazard are unreliable, as there are few of such cases. The consequence is that predictions suggest more precision than can be provided by the knowledge available, which may lead to wrong decisions.

7.3 IMPRECISE PROBABILISTIC HORN CLAUSE LOGIC

We first introduce the idea behind IPHL and then formally develop its semantics.

7.3.1 *Basic Idea*

Imprecise probability intervals can be seen as a way to unify the qualitative approach to deal with uncertainty of alethic model logic and the quantitative approach of probabilistic LP. The basic idea of IPHL is therefore to extend LP with probability interval annotations, such as probabilistic LP extends LP with point probabilities. Our language supports two possible interpretations of rules as:

$$\mathbf{p}\colon h \leftarrow b_1, \ldots, b_n,$$

where $\mathbf{p}$ is a probability interval, which can be closed, open, or half-closed. We refer to those different interpretations as different kinds of *imprecisions*: *rule-imprecisions* and *head-imprecisions*. The kind of imprecision determines how probabilities of multiple rules with the same head are combined. Formally, we say an IPHL program $\mathbf{T} = (\mathbf{R}_R, \mathbf{R}_H)$, consists of rules with rule-imprecisions $\mathbf{R}_R$ and rules with head-imprecisions $\mathbf{R}_H$. The heads occurring in $\mathbf{R}_R$ and $\mathbf{R}_H$ are disjoint.

Rules in $\mathbf{R}_R$ are denoted as:

$$\mathbf{p}\colon (h \leftarrow b_1, \ldots, b_n),$$

with $\mathbf{p}$ a probability interval. The interpretation of these rules is that the probability that $b_1, \ldots, b_n$ leads to h is in $\mathbf{p}$, which corresponds to the semantics of precise probabilistic programming as given in Section 7.2.3. Multiple rules with the same head are combined using a noisy-or operator for probability intervals.

Example 7.5

Consider an imprecise version of Example 7.3:

$$\begin{aligned}
&[0.05, 0.15]\colon (\mathit{env\_hazard} \leftarrow \top) \\
&[0.4\ \ , 0.6\ \ ]\colon (\mathit{env\_hazard} \leftarrow \mathit{chemicals})
\end{aligned}$$

This means that carrying chemicals causes a vessel to be an environmental hazard with probability between 0.4 and 0.6, while it is unlikely that other reasons cause a ship to be an environmental hazard.

In case a vessel has no chemicals on board the probability of it being an environmental hazard is between 0.05 and 0.15. Otherwise, we consider the probabilities one gets for all possible choices of probabilities from the intervals given the semantics of point probabilities (Section 7.2.3). These probabilities are within the interval $[0.43, 0.66]$.

Rules in $\mathbf{R}_H$ are denoted as:

$$(\mathbf{p}\colon h) \leftarrow b_1, \ldots, b_n$$

As indicated by the brackets, in this case, the probability interval only applies to the head. The rules are interpreted as follows: in case $b_1, \ldots, b_n$ holds, the probability of h is in $\mathbf{p}$. This means that rules do not represent independent causes for the head, but conditions under which the knowledge about the head's probability becomes more precise. If no rule applies, there is complete ignorance about the head's probability, i.e. it is in $[0.0, 1.0]$. Note that this kind of rules can lead to inconsistent definitions, whereas rules with rule-imprecisions cannot. While the interpretation of head-imprecisions makes no sense for the precise case, it can conveniently express certain kinds of imprecise knowledge.

Example 7.6

Suppose we have statistical knowledge about tankers, for example because all tankers have to register their cargo due to safety regulations, and can estimate the likelihood of tankers having chemicals loaded as 0.3. For all other vessels, we do not have that information, because we do not have data about all ships, and only express that it is possibly carrying chemicals. We interpret this as the probability interval $(0.0, 1.0]$, i.e. the probability is greater 0.0. This knowledge can be modelled with head-imprecisions:

$$\begin{aligned}&\big((0.0, 1.0]\colon \mathit{chemicals}\big) \leftarrow \top\\ &\big([0.3, 0.3]\colon \mathit{chemicals}\big) \leftarrow \mathit{tanker}\end{aligned}$$

Given the rules above, if we do not know that the ship is a tanker, its probability of having chemicals on board is in $(0.0, 1.0]$. In the case it is a tanker, it is in $(0.0, 1.0]$ and in $[0.3, 0.3]$, which means it is in $(0.0, 1.0] \cap [0.3, 0.3] = [0.3, 0.3]$.

7.3.2 *Qualitative Interpretation*

Given the basic language defined above, we can give various intervals a qualitative interpretation. For example, special cases of intervals include *determinism* ($[0.0, 0.0]$ and $[1.0, 1.0]$), complete ignorance as in 3-valued logics ($[0.0, 1.0]$) and precise probabilities ($[p, p]$ with p some probability).

IPHL can also be seen as a basic language to define various modalities in terms of probability intervals. For example, the interval $[1.0, 1.0]$ corresponds to the modality that something is necessarily true: $\Box$. Furthermore, the modality

$\Diamond$, expressing that some statement is possible, in its least strict interpretation, means that the probability is greater than 0.0, i.e. it is in the interval $(0.0, 1.0]$. Analogously, the modality $\Diamond\neg$ corresponds to the interval $[0.0, 1.0)$. One could alternatively more carefully only consider statements possible in case their probability is above a certain threshold and define for instance $\Diamond_{0.05}$ as $(0.05, 1.0]$. Also linguistic modalities, such as *unlikely* with the possible meaning of $[0.05, 0.15]$, can be introduced in the framework of IPHL. Note that in order to differentiate between complete ignorance ($[0.0, 1.0]$) and possibility ($(0.0, 1.0]$) the distinction between open and closed intervals is necessary, which cannot be made in commonly used imprecise formalisms, such as LDCNs [31].

Similar to a qualitative specification of the IPHL program, probability intervals of queries can also be given a qualitative interpretation, which implies that the language supports qualitative reasoning as a special case. For example, a probability interval of $[1.0, 1.0]$ means a particular statement is necessarily true and a probability greater 0.0 implies that the statement is possible. Furthermore, given the lower bound of a probability interval, we may conclude whether or not the probability is possibly or necessarily larger than a given threshold. Finally, one can also qualitatively compare the likelihood of two statements. For instance, in case the probability intervals of two statements are disjoint, one can determine which one is more likely than the other.

7.3.3 *Semantics*

In accordance with probabilistic logics programming, the semantics of IPHL programs is defined in terms of the *marginal* probability intervals of arbitrary query *atoms*. The semantics is defined incrementally, by first extending the semantics for precise logic programs given in Section 7.2.3 for rules with rule-imprecision. Then, we also show how to deal with head-imprecisions.

7.3.3.1 *Rules with Rule-Imprecisions*

We first develop a semantics, assuming the program only consists of rules with rule-imprecisions. The semantics of rules with rule-imprecisions is defined in terms of a set of programs obtained by replacing the intervals with point probabilities. In other words, we consider all programs for all possible choices of probabilities for each interval.

Definition 7.1 (Equivalent Set of Programs for Rule-Imprecisions). *Let $\mathbf{R}_R$ be a set of rules with rule-imprecisions. Then, the equivalent set of programs $\mathcal{T}$ contains all possible probabilistic logic programs, in which for each rule $\mathbf{p}\colon (h \leftarrow b_1, \ldots, b_n)$ in $\mathbf{R}_R$ a rule $p\colon (h \leftarrow b_1, \ldots, b_n)$ is included, such that $p \in \mathbf{p}$.*

We can then define the probability interval of a query q given a program.

Definition 7.2 (Query Probability with Rule-Imprecisions). *Let $\mathcal{T}$ be the equivalent set of programs of a set of IPHL rules $\mathbf{R}_R$. The probability of a query q given such IPHL theory is:*

$$P(q) \overset{\text{def}}{\in} \{P_{\mathbf{R}}(q) \mid \mathbf{R} \in \mathcal{T}\},$$

where $P_{\mathbf{R}}(q)$ is defined as in Equation 7.2.

This set of probabilities is convex and can therefore be expressed as interval.

Example 7.7

Consider again the imprecise program of Example 7.5:

$$\begin{aligned}&[0.05, 0.15] \colon (\mathit{env\_hazard} \leftarrow \top)\\&[0.4\ \ , 0.6\ \ ] \colon (\mathit{env\_hazard} \leftarrow \mathit{chemicals})\end{aligned}$$

An example of an element of $\mathcal{T}$ is:

$$\begin{aligned}&0.1 \colon \mathit{env\_hazard} \leftarrow \top\\&0.4 \colon \mathit{env\_hazard} \leftarrow \mathit{chemicals}\end{aligned}$$

Actually, all uncountably many programs with the following structure form $\mathcal{T}$:

$$\begin{aligned}&p \colon \mathit{env\_hazard} \leftarrow \top\\&q \colon \mathit{env\_hazard} \leftarrow \mathit{chemicals}\end{aligned}$$

where $0.05 \leq p \leq 0.15$ and $0.4 \leq q \leq 0.6$.

Then $P(\mathit{env\_hazard}) \in [0.05, 0.15]$, since there is no rule with head *chemicals*. Therefore, the probability of *chemicals* is 0.0 and the second rule never applies. In case we assume, that we know that the vessel is carrying chemicals and add:

$$[1.0, 1.0] \colon (\mathit{chemicals} \leftarrow \top),$$

the probability for *env_hazard* is in $[0.43, 0.66]$. Those bounds correspond to the noisy-or combination for the bounds of both rules.

7.3.3.2 *Rules with Head-Imprecisions*

Next, we extend the semantics with head-imprecisions. Probability intervals on heads have a different characteristic than probability intervals on rules. Head-imprecisions can be looked upon as constraints that exclude programs for which the probability distribution does not obey the specified bounds, in contrast to rules which rule-imprecisions, which generate programs. Therefore, we first generate rules allowing for all possible probabilities and then enforce the constraints on the set of programs. To allow all possible probabilities for rules with head-imprecision, we add for each head h occurring in the rules $\mathbf{R}_H$ a rule $[0.0, 1.0] \colon (h \leftarrow \top)$ to $\mathbf{R}_R$ and call the resulting set of rules $\mathbf{R}'_R$.

We can now consider a set of programs $\mathcal{T}$ with point probabilities generated by $\mathbf{R}'_R$, under the semantics of rule-imprecisions. To incorporate the constraints, given by the probability intervals on heads, we define a set of programs $\mathcal{T}'$ by including only those programs obeying all constraints given by $\mathbf{R}_H$.

Definition 7.3 (Equivalent Set of Programs for General IPHL Theory)**.** *Given an equivalent set of programs $\mathcal{T}$ generated by $\mathbf{R}'_R$ constructed from an IPHL theory $\mathbf{T} = (\mathbf{R}_R, \mathbf{R}_H)$, the set of programs $\mathcal{T}'$ equivalent to $\mathbf{T}$ is:*

$$\mathcal{T}' \overset{\text{def}}{=} \{\mathbf{R} \in \mathcal{T} \mid \underset{(\mathbf{p}:\, h)\leftarrow b_1,\ldots,b_n \in \mathbf{R}_H}{\forall} P_{\mathbf{R}}(h \mid b_1, \ldots, b_n) \in \mathbf{p}\}$$

In case $\mathcal{T}'$ becomes the empty set, the entire program is called *inconsistent*. For consistent theories, the probability of a query q is defined as in Definition 7.2 using $\mathcal{T}'$ instead of $\mathcal{T}$.

Example 7.8

As example, we use the following program, which is a combination of the rules of Examples 7.5 and 7.6:

$$\begin{aligned}\mathbf{R}_R = \{ \quad & [0.05, 0.15]: (\mathit{env\_hazard} \leftarrow \top) \\ & [0.4, \ \ 0.6\ ]: (\mathit{env\_hazard} \leftarrow \mathit{chemicals}) \ \} \\ \mathbf{R}_H = \{ \ & \big((0.0, \ \ 1.0\ ]: \ \mathit{chemicals}\big) \ \leftarrow \top \\ & \big([0.3, \ \ 0.3\ ]: \ \mathit{chemicals}\big) \ \leftarrow \mathit{tanker} \ \}\end{aligned}$$

From this we get the following transformed set of rules:

$$\begin{aligned}\mathbf{R}'_R = \{ \ & [0.05, 0.15]: (\mathit{env\_hazard} \leftarrow \top) \\ & (0.4, 0.6 \quad]: (\mathit{env\_hazard} \leftarrow \mathit{chemicals}) \\ & [0.0, 1.0 \quad]: (\mathit{chemicals} \quad \leftarrow \top) \ \}\end{aligned}$$

The equivalent set of programs $\mathcal{T}'$ given the set of programs $\mathcal{T}$ generated by $\mathbf{R}'_R$ is then:

$$\mathcal{T}' = \{\mathbf{R} \in \mathcal{T} \mid P_{\mathbf{R}}(\mathit{chemicals} \mid \mathit{tanker}) = 0.3 \wedge 0.0 < P_{\mathbf{R}}(\mathit{chemicals})\}$$

From this we can conclude that the probability for *env_hazard* is strictly greater than 0.05. This is because the probability of *chemicals* is above 0.0, which causes *env_hazard* with a probability above 0.0, meaning that a non-zero probability is added to the 0.05 originating from the first rule. In case we assume *tanker*, the probability for *chemicals* is 0.3. Then $P(\mathit{env\_hazard}) \geq 1 - (1 - 0.05) \cdot (1 - 0.4 \cdot 0.3) = 0.164$, which is obtained by computing the noisy-or combination.

7.4 INFERENCE

In this section, an inference mechanism is introduced, which translates the problem of computing the lower or upper bound respectively to a precise probabilistic inference problem without changing inference complexity. We only deal with exact inference, as approximate inference could change the qualitative nature of the answer. For instance, there is a qualitative difference of a probability greater zero and greater or equal to zero, which cannot be made by sampling algorithms. Similarly to the semantics, we introduce the inference method incrementally, starting with point probabilities.

7.4.1 *Rules with Point Probabilities*

We already discussed probabilistic inference by WMC in Sections 5.1 and 5.2.1. Here we however introduce a concrete translation to a WMC problem, which serves as a basis for the inference method for IPHL. The issue here is that WMC is defined for propositional *knowledge bases* without the closed world assumption. The rules therefore have to be translated to propositional logic by making the head literals equivalent to the disjunction of all rules defining it to capture the closed world assumption. This is known as *Clark's completion* [26] and a similar approach for probabilistic inference is taken by *ProbLog* [48].

Probabilities are added by introducing auxiliary atoms for each rule. So in the first step each rule $R = p\colon (h \leftarrow b_1, \ldots, b_n)$ is translated to $h \leftarrow aux_R, b_1, \ldots, b_n$, before Clark's completion is performed. The weight p is assigned to aux_R and $1 - p$ to its negation. All other literals get weight 1. We denote the resulting weighted knowledge base with $\Delta_{\mathbf{R}}$. The probability can then be computed as $P_{\mathbf{R}}(q) = \text{WMC}(\Delta_{\mathbf{R}} \wedge q)$.

Example 7.9

Consider the following rules:

$$0.1\colon env\_hazard \leftarrow \top$$
$$0.4\colon env\_hazard \leftarrow chemicals$$

We furthermore assume that the vessel is carrying chemicals ($1.0\colon chemicals \leftarrow \top$). The resulting weighted knowledge base $\Delta_{\mathbf{R}}$ is:

$$\big(env\_hazard \leftrightarrow aux_1 \vee (aux_2 \wedge chemicals)\big) \wedge chemicals$$

Note that for brevity we added *chemicals* as fact instead of making it equivalent to an auxiliary literal with weight 1.0. The weights are: $w(aux_1) = 0.1$, $w(\neg aux_1) = 0.9$, $w(aux_2) = 0.4$, $w(\neg aux_2) = 0.6$, $w(env\_hazard) = w(\neg env\_hazard) = 1.0$, $w(chemicals) = 1.0$ and $w(\neg chemicals) = 0.0$.

To compute the probability of *env_hazard*, we consider the models of the knowledge base in which *env_hazard* holds, with corresponding weights, computed as the product of all weights of literals included:

$$0.04\colon env\_hazard,\ \ aux_1,\ \ aux_2, chemicals$$
$$0.06\colon env\_hazard,\ \ aux_1, \neg aux_2, chemicals$$
$$0.36\colon env\_hazard, \neg aux_1,\ \ aux_2, chemicals$$

The sum of those weights is 0.46, which corresponds to the probability according to the semantic definition as illustrated in Example 7.4.

7.4.2 *Imprecise Inference*

Given the fact that we only make use of intervals, the set of probabilities of a query is always convex. We can therefore represent the semantically infinite set of probabilities by its extreme points. We denote the lower and upper bounds of the result probability as $\underline{P}(q)$ and $\overline{P}(q)$ respectively. The basic idea of the inference algorithm is to translate the problem to a precise probabilistic inference problem, for both bounds. An algorithm for the lower bound is given in Algorithm 4. Inference for the upper bounds can be done analogously.

Algorithm 4: Imprecise Inference (lower bound)

```
   Input: Query $q$ and IPHL program $(\mathbf{R}_R, \mathbf{R}_H)$
   Result: The lower probability bound of $q$
1  $\mathbf{R} = \emptyset$
2  for $(\mathbf{p}\colon (h \leftarrow b_1, \ldots, b_n)) \in \mathbf{R}_R$
3      $\mathbf{R} \hookleftarrow (lower(\mathbf{p})\colon (h \leftarrow b_1, \ldots, b_n))$
4  for all heads $h$ in $\mathbf{R}_H$
5      for $\{b_1, \ldots, b_n\} \subseteq \mathbf{B}$, with $\mathbf{B}$ all body atoms defining $h$
6          $\mathbf{R} \hookleftarrow (subset\_lower_h(\{b_1, \ldots, b_n\})\colon (h \leftarrow b_1, \ldots, b_n))$
7  return WMC$(\Delta_{\mathbf{R}} \wedge q)$
```

7.4.2.1 *Inference for Rule-Imprecisions*

The fact that we use Horn clauses, thus bodies consisting of positive atoms only, makes it possible to locally determine the extreme points of each rule independent of the query. In fact, choosing the minimum or maximum probability for all rules determines the minimal or maximal probability for all possible queries, respectively. This is the key insight which makes the inference problem as complex as the precise counterpart: without the restriction to Horn clauses, the combination of all rules' extreme points would have to be considered. This requires considering an exponential number of precise probabilistic programs, which increases the complexity of the inference problem. For PCLP, elimination of *imprecise constraints* possibly has to be postponed during inference, as it cannot be decided to which value of related *random variables* the imprecise probability mass has to be assigned in order to minimize or maximise the probability of the query. This increases parametrised complexity of PCLP inference. For IPHL we do not have to postpone that decision, as assigning the imprecise probability mass to *true* maximises all probabilities and assigning the mass to *false* minimises them. Therefore, for the IPHL rules with rule-imprecision the translation for computing the lower bound is done by for each rule using its lower bound probability (Algorithm 4, Lines 2, 3).

In order to represent open as well as closed intervals, we make use of a calculus for *hyperreal* numbers [56]. For instance, the interval $(0.2, 0.7)$ can be represented by the extreme points $0.2 + \mathrm{d}$ and $0.7 - \mathrm{d}$, where d represents an infinitesimal number. WMC can be extended with hyperreal weights, by making use of addition and multiplication as defined for the hyperreal calculus. In Algorithm 4, the function $\mathit{lower}(\mathbf{p})$, is defined as $\min(\mathbf{p})$ in case the lower bound is closed and $\sup(\mathbf{p}) + \mathrm{d}$ otherwise.

Example 7.10

Consider the following rules:

$$\begin{aligned}
(0.1,\ 0.2\ )&: (\mathit{chemicals} \leftarrow \top)\\
[0.05, 0.15]&: (\mathit{env\_hazard} \leftarrow \top)\\
[0.4,\ 0.6\ ]&: (\mathit{env\_hazard} \leftarrow \mathit{chemicals})
\end{aligned}$$

For each query the lower bound of the probability is the probability given the following precise probabilistic logic program:

$$\begin{aligned}
0.1 + \mathrm{d}&: \mathit{chemicals} \leftarrow \top\\
0.05&: \mathit{env\_hazard} \leftarrow \top\\
0.4&: \mathit{env\_hazard} \leftarrow \mathit{chemicals}
\end{aligned}$$

Lemma 7.1. *For any IPHL program $\mathbf{T} = (\mathbf{R}_R, \varnothing)$ and any query q, Algorithm 4 computes the correct probability lower bound of q, according to Definition 7.2.*

Lemma 7.2. *For any IPHL program $\mathbf{T} = (\mathbf{R}_R, \varnothing)$, the computational complexity of computing any query is the same as for a probabilistic logic program consisting of $\mathbf{R}_R$, in which all intervals are replaced by point probabilities.*

In practice, inference could even be less expensive if the extremes of some intervals are 0.0 and 1.0, introducing additional determinism, which can be exploited by WMC algorithms.

7.4.2.2 *Inference for Head-Imprecisions*

The translation of rules with head-imprecision is more involved (Algorithm 4, Lines 4 – 6). For each head h (Line 4) we have to consider all cases of truth assignments to all atoms in the bodies of the rules defining h (Line 5). Each of those cases corresponds to a subset $\mathbf{B}$ of those atoms. We can define the probability interval for h by the intersection of all intervals of rules which apply given such $\mathbf{B}$:

$$\mathbf{P}_h(\mathbf{B}) \overset{\text{def}}{=} \bigcap_{\substack{\{b_1,\ldots,b_n\} \subseteq \mathbf{B}\\ (\mathbf{p}:\, h) \leftarrow b_1,\ldots,b_n \in \mathbf{R}_h}} \mathbf{p}, \tag{7.3}$$

where $\mathbf{R}_h$ are all rules with head h. $\mathbf{P}_h(\mathbf{B})$ is never the empty set for consistent programs.

Naively translating to rules with all possible $\mathbf{B}$ as body and using the lower bounds of such intervals, i.e. $\mathit{lower}(\mathbf{P}_h(\mathbf{B}))\colon (h \leftarrow \mathbf{B})$, results in incorrect probabilities, since all rules with a subset of $\mathbf{B}$ as body also contribute to the probability of h in case $\mathbf{B}$ holds. To solve that problem we make use of the property that with increasing cardinality of $\mathbf{B}$, the number of satisfied bodies only increases. Therefore with increasing cardinality of $\mathbf{B}$ the probability is restricted to a more tight interval, as it is restricted to the intersection of all such rules' intervals. This implies that the lower bound monotonically increases with the cardinality of $\mathbf{B}$.

For the correct transformation of the probabilities we use in Line 6 of Algorithm 4 the function *subset_lower*, which computes the correct probabilities for each subset $\mathbf{B}$. The idea is to consider the probability already given by the rules corresponding to proper subsets of $\mathbf{B}$ and only add as much probability mass as is needed to get the desired lower bound of the probability $\mathbf{P}_h(\mathbf{B})$:

$$\mathit{subset\_lower}_h(\mathbf{B}) \overset{\text{def}}{=} 1 - \frac{1 - \mathit{lower}(\mathbf{P}_h(\mathbf{B}))}{\prod\limits_{\mathbf{B}' \subset \mathbf{B}} 1 - \mathit{subset\_lower}_h(\mathbf{B}')}, \tag{7.4}$$

where the denominator is 1 for the empty set.

Proposition 7.1. *For any IPHL theory* $\mathbf{T}$ *and any query* q*, Algorithm 4 computes the correct lower probability bound of* q*, as defined by Definitions 7.2 and 7.3.*

Example 7.11

Consider the following rules with head-imprecision:

$$\big((0.0, 1.0]\colon \mathit{chemicals}\big) \leftarrow \top$$
$$\big([0.3, 0.3]\colon \mathit{chemicals}\big) \leftarrow \mathit{tanker}$$

For inference they are translated to the following rules with rule-imprecision:

$$\begin{aligned} &\mathrm{d} &&\colon (\mathit{chemicals} \leftarrow \top) \\ &1 - (0.7/(1-\mathrm{d}))&&\colon (\mathit{chemicals} \leftarrow \mathit{tanker}) \end{aligned}$$

In case *tanker* holds, the probability of *chemicals* according to Equation 7.2 is $(1 - (0.7/(1-\mathrm{d})))\mathrm{d} + (0.7/(1-\mathrm{d}))\mathrm{d} + (1 - (0.7/(1-\mathrm{d})))(1-\mathrm{d}) = 0.3$, which equals the lower bound for that case, as defined by the rules above.

A similar transformation of rules with head-imprecision is incorrect for the upper bound, because the upper bound decreases with larger cardinality of $\mathbf{B}$ and additional rules of which the body holds can only increase, but not decrease, the probability. This problem is solved by actually computing the lower bound of the query's negation and computing the upper bound of the original query as $\overline{P}(q) = 1 - \underline{P}(\neg q)$. To compute the lower bound of the negation we transform the knowledge base, such that each atom actually represents its negation. This can be achieved by swapping $\wedge$ and $\vee$ in Clark's completion and use as weight for auxiliary atoms $1 - upper(\mathbf{p})$.

To characterize the computational complexity of a full IPHL program, it makes no sense to speak about a corresponding precise probabilistic logic program for IPHL programs with head-impressions as in Lemma 7.2, since it makes no sense to replace all probabilities with point probabilities for such rules. However, we still get the following guarantee in terms of complexity of programs with a similar structure.

Theorem 7.1. *Inference for a IPHL programs has the same complexity in terms of the treewidth, as a corresponding precise probabilistic logic program, in which each head is defined in terms of the same body atoms as in the IPHL program.*

7.5 RELATED WORK

Several approaches have been proposed to combine probability theory and qualitative modalities. One example is the logic of Bacchus [6], which makes it possible to put constraints on probabilities as 'the agent believes φ with probability greater than 0.5'. The connection to imprecise probability theory is however not made in this work. There are also languages allowing one to express higher-order probabilistic statements [64], such as 'the probability that the probability of φ is larger 0.5 is 0.9'. This work is mainly theoretical in nature and efficient inference mechanisms are not provided.

The epistemic logic languages of Milch and Koller [101] and Shirazi and Amir [143], deal with beliefs of multiple agents and also higher-order beliefs about beliefs, and therefore have a different goal than our approach. They provide mechanisms for exact inference, based on *Bayesian networks* (BNs). However, in their work probabilistic inference is only a subroutine of the complete inference method whereas the complete inference is strictly more complex than ordinary probabilistic inference.

There is some work on inference for imprecise formalisms such as LDCNs [31]. All exact approaches, such as [20, 47], suffer from the worst-case complexity of the problem. We do not discuss approximate methods here, since they are not suited for qualitative inference, as we discussed before.

7.6 CONCLUSIONS

We introduced an imprecise probabilistic logic language based on Horn clauses, which makes it possible to express and unambiguously define qualitative statements with varying level of precision, with as special cases complete ignorance, point probabilities and determinism. This is made possible by a solid semantic foundation based on imprecise probability theory. We have furthermore shown that it is possible to provide inference for an imprecise language, which is as expensive as its precise counterpart, while in general imprecise inference problems are more complex. Also, as the work on PCLP and GWMC in this thesis, the result shows that it is possible to employ state-of-the-art probabilistic inference methods for imprecise problems.

8 APPLICATION TO MARITIME SAFETY AND SECURITY TASKS

This chapter discusses an application of *Probabilistic Constraint Logic Programming* (PCLP) to the realistic problem of maritime safety and security.

8.1 AUTOMATED SUPPORT FOR MARITIME SAS TASKS

We present a particularly challenging application for which probabilistic reasoning, beyond the *propositional* case, is essential and show that *probabilistic logics* and in particular PCLP is very suited to tackle the challenges emerging from the application domain. The domain we consider are maritime *safety and security* (SaS) tasks, involving the continuous monitoring of vessels to detect and predict events such as accidents or illegal activities.

A great deal of research in the past has been devoted to fusing sensor data to track object motion [52, 59]. Measurements of quantities, such as positions, which are relevant for this task, are available from sensors under own control with known range and error characteristics. For SaS tasks, however, the ultimate goal is to provide situational awareness, which requires the identification of monitored objects and the understanding of their behaviour. This is a challenging and complex problem for human operators as they need to deal with:

- **Large amount of monitored objects:** There is a huge and increasing number of vessels active in the world. For instance, around 2 000 ships are active daily off the coast of the Netherlands alone. There are additionally many small boats, which are hard to detect and often used for illegal activities. Moreover, considering other involved parties such as companies, insurances and persons is essential.

- **Rare and diverse relevant behaviour:** Only very few vessels show behaviour that requires action. Furthermore, the nature of such behaviour is very diverse, so there is not a single indicator for relevant behaviour. Examples of such behaviour include environmental pollution, smuggling and behaviour increasing the risk of accidents. Moreover, perpetrators often do their best to make their behaviour look as much as possible like the harmless behaviour of the majority of other vessels.

- **Huge amount of uncertain data:** There is a huge amount of data available, which is however very heterogeneous, as each source represents information in different formats and at various levels of detail. Additionally, data comes with different kinds of uncertainties, such that it cannot straightforwardly be used to detect relevant behaviour. The most severe cause of uncertainty is that, due to the limited sensor placement at sea, directly observable information is very limited. One can neither observe relevant behaviour directly, nor can be sure of objects' identities, which is essential to link objects with other information available. The data available is furthermore of varying quality, because data comes from diverse sources such as ship transmitters, websites or commercial databases. Data from the various sources all contain erroneous information to different extends, either accidentally, intentionally or because data is outdated.

The main consequence of the characteristics of the domain is that one has to accept and be able to deal with uncertainty to an extend beyond what is required for traditional sensor fusion. High certainty can only be achieved by close inspection, which is too costly to do for all vessels, so the goal is to inspect those vessels with the highest chance of behaviour which is relevant for the current SaS task's goal. Such likelihood estimates have to be obtained by using indirectly related information. For example, even if one cannot determine the identity with high certainty, one could draw conclusions based on a number of possible identities. Uncertainty is reduced most by using as much relevant data as possible. The more uncertain the data is, the more data is required to come to good estimates.

Summarised, the maritime SaS task requires to reason about an amount of uncertain information, that is as large as possible, at the same time has complex relations and is only indirectly related to the facts reasoned about. This has to be done for a huge number of ships simultaneously. For humans this leads to *information overload*, with the risk of missing obviously relevant behaviour. This motives why automated support is needed. Note that the aim of such support systems currently can only be to support the operator to focus on potentially interesting vessels, but cannot replace decision making based on human judgement.

Automating support of this task is very challenging. A support system has to coordinate complex tasks such as collecting data, deciding when and how to reason about which behaviour of which vessels and communicating the results to the user. Reasoning about the information available is however a central task and we focus on this aspect. The most important requirements for the reasoning component are:

- **Data and knowledge-based model:** The reasoning model must be based on a combination of data and knowledge, which excludes pure machine learning methods. Data directly related to the aims of reasoning is rare, as discussed above. There may be enough data to build a model of how the category of a vessel relates to its tonnage, but data that relates behaviour to other factors is hardly available. That part of the model can therefore only be built by making use of expert knowledge. Also new insights may have to be quickly incorporated into the model, with no time to first collect enough data, as strategies of criminals often change quickly.

- **Relations between dynamic numbers of objects:** As there is a lot of dynamics in the domain, the model must express complex relations between a non-fixed number of objects. It is not known in advance how many pieces of possibly relevant information is found or how many possibly identities have to be considered. So the modelling language employed must be expressive enough to support formalising general knowledge, applicable to a dynamic number of objects.
- **Quantitative uncertainty reasoning:** The method employed has to be able to deal with uncertainty in a quantitative way. There are very few facts, which are known for sure and very likely contradictions between pieces of information. The consequence is that most kinds of formalisms, not being able to deal with quantitative uncertainty, fail. One also wants to compare in a fine-grained way how much chance of success a certain action has on different vessels, which is not possible based on qualitative judgements.
- **Hybrid reasoning:** The method employed has to be able to deal with discrete as well as continuous variables, because in this domain continuous factors are too important to be ignored. All information that is initially known about an object at sea with high certainty is based on radar data and is numeric. Concretely, the speed and dimensions of the object are usually known and can e.g. already show that it is very unlikely that the pretended identity of an object is the actual one.
- **Exposing internal reasoning information:** The reasoning system should expose as much information about its internal state as possible. Such information is very important for the working of the complex system in which the reasoner is embedded, also including human experts. There are several important aspects information should be exposed about:
  - **Reasoning status & result quality:** It is hard to estimate how long computing probabilities takes given a collection of information, due to the complexity characteristics of the problem. Small changes to the input or the model can lead to orders of magnitude different computation times. Anytime algorithms, producing intermediate approximations, can solve that problem, but only in case they provide information about the quality of approximations.
  - **Structure of the model:** The structure of the model and therefore also of the knowledge should not be a black box. This is important to provide to the reasoner information, which is most relevant for the current reasoning goal.
  - **Conclusions rationale:** It is very important for human operators to get their own understanding of the situation before they take serious

action. To what extend this is possible influences acceptance of the entire system (see also [19, 90]). So the reasoner should expose how it came to its conclusions, which should be double checked by human experts, as the model is necessarily limited and only meant as support. Finally, understanding how the reasoner reached incorrect conclusions, is very valuable for improving the model.

While recent systems, developed e.g. in projects such as *Poseidon*[1] and *DeMarine*[2], allow automatic identification of abnormalities, they lack capabilities for real-time prioritisation of the application tasks, selection and alignment of relevant information, and efficient reasoning at a situation level. Such intelligent capabilities are embedded in the innovative system developed within the *METIS* project [65, 151] based on the employment and integration of state-of-the-art *artificial intelligence* (AI) approaches. In this chapter, we describe in particular the application of PCLP to address the aforementioned challenges for automated reasoning for maritime SaS tasks. Concretely, we discuss a model we employed as essential part of the METIS prototype system and especially focus on why PCLP is especially suited for building and integrating such model.

We focus on the first requirements stated above, which are related to modelling. *Inference* related issues were already discussed in this thesis before. The inference method introduced in Chapter 6 is particularly suited for automated reasoning in the this context, as the current maximal error is guaranteed and thus one can judge whether the current approximation produced is good enough to take a decision. PCLP models also make it possible to expose the structure of knowledge and reasoning. The details are out of the scope of this thesis, but in other work we show that information derived from PCLP models can be used to dynamically reconfigure the information retrieval process based on the current mission [151]. Similarly, we show elsewhere, in collaboration with visualisation experts, that it is possible to make visible for human experts how the reasoning systems came to its conclusions [151, 141].

In this chapter we first discuss some domain knowledge in detail (Section 8.2) and then introduce a probabilistic PCLP model, which we built for supporting the SaS task in Section 8.3. We then, in Section 8.4, evaluate the model's performance qualitatively and quantitatively and also show applicability in the context of a complex support system. Finally, related work is discussed in Section 8.5 and the chapter is concluded in Section 8.6.

8.2 MARITIME SAFETY AND SECURITY DOMAIN KNOWLEDGE

We discuss the characteristics of the domain and the knowledge we used to build the model. First, we give some qualitative knowledge, which we obtained from freely available sources and the collaboration with our industrial partner

1 http://www.esi.nl/poseidon
2 http://www.ceon-bremen.de/Maritime_Safety_and_Security

and then some quantitative statistical knowledge obtained from data and expert knowledge.

8.2.1 *Qualitative Knowledge*

Identifying a vessel is not straightforward. Traditionally vessels have a name, painted on its hull, which is however not a perfect identifier, as it is not unique. Also vessels change their names, usually in case they are sold to another owner. Still, the name is very relevant as it is often mentioned in news articles and can visually be read from the hull. A better identifier is the *Maritime Mobile Service Identity* (MMSI), which identifies ships, but also other radio stations e.g. at the coast, in digital communication. A desirable property is that MMSIs are unique at least at one point in time, but they are not stable over the entire vessel's lifetime. It actually identifies the radio station rather than the vessel itself, and is therefore comparable to a telephone number of a person, which can not only change, but can also be reassigned to a different person after some time. The vessel's flag is also encoded in this number, so a flag change always results in a changed MMSI as well. The only unique and stable identifier is the *International Maritime Organization* (IMO) number, which is however only assigned to large vessels, concretely to cargo vessels with at least 300 gross tons and passenger vessels of at least 100 gross tons.

Another important property of vessels is their category, of which the most important ones are tankers, container, passenger and fishing vessels. Obviously, the category is related to the physical properties such as size, gross tonnage and maximum speed, but also to vessels' behaviour. The category furthermore determines to what extend a vessel poses a threat in the current context. For instance, a tanker is a potential threat particularly in an environmentally sensitive area. Additionally, as with SaS tasks in general, in the maritime domain the behaviour of the objects of interest plays a crucial role in detecting abnormal *events*. It is known, for example, that vessels being involved in illegal activities, such as smuggling or hijacking, may try to hide their identity by repainting the vessel on sea.

Knowledge of the characteristics of information provided by various sources, is in this domain at least as important as knowledge about vessels themselves. Radars detect only the existence of objects and their movement, but provide no information about the objects' identities. The most accessible source for that extra information is the *Automatic Identification System* (AIS). It is an international standard for communication of vessel information, including important identity attributes such as the MMSI, the IMO number, the name and the category. Currently, most vessels broadcast this information using the AIS system. Despite its wide availability, AIS information is unreliable. A ship can send erroneous information or no AIS signal at all, especially if the ship has a reason to hide its identity. There is a substantial number of vessels that provide incomplete information, e.g. they provide no names, or information contradicting with other sources in their AIS messages. We confirmed this by analysing a dataset of logged AIS messages sent by vessels in the Dutch *exclusive economic zone*, which was provided by the *Maritime Research Institute Netherlands* (MARIN). However, information provided by AIS can be evaluated and enriched by using information from additional sources such as *IHS Fairplay*[3], a commercial database containing detailed vessel information, and *MarineTraffic*[4], a free-to-use website that provides real-time ship tracking information. We found many contradictions of the data from these sources with AIS data, e.g. there were different names for the same MMSI, but also contradictions between those additional sources.

The way experts judge the quality of information is not only determined by the sources, but also by the intentions of involved parties. For instance, a vessel hiding its identity, more likely transmits incorrect information via AIS. So *evidence* about behaviour and intentions influence how information is judged and vice versa, meaning that the reasoning task involves weighing many factors.

8.2.2 *Quantitative Knowledge*

As discussed, quantitative, statistical knowledge about behaviour and intentions of vessels is barely available and we have to rely on expert estimates. Still, we used the discussed AIS dataset to also get some statistical knowledge about physical attributes, as category and tonnage. Even though certain errors exist in AIS data, we assume that these errors do not significantly bias estimates, e.g. though some ships may have wrongly reported categories, the overall distribution of categories can be estimated from the data. We used data of three days, each including about 2 000 vessels and restrict ourselves to the MMSI, the IMO, the name and the category in the following. We found out that about half of the vessels have no IMO and the number of names is about 80% of the number of vessels. For the category of vessels we estimated the following distribution: $\{0.4 : \mathit{cargo}, 0.2 : \mathit{tanker}, 0.05 : \mathit{passenger}, 0.05 : \mathit{fishing}, 0.3 : \mathit{other}\}$. We also obtained distributions for numeric attributes, such as the gross tonnage and maximal speed, for the different categories, which we however not use in this chapter. The total number of used MMSIs could not be estimated using AIS data of three days only, but we used the total number of active ships – 380 000 – as an estimation[5]. The error rates of the sources used cannot directly be estimated from the data since the ground truth is unknown, but we use estimates based on the knowledge of maritime experts: the error rate of AIS is 5% in case a vessel is not hiding its identity, 1% for IHS Fairplay, 2% for websites and visual observations are completely trusted.

8.2.3 *Maritime Example Scenario*

A coast guard operator has obtained an intelligence report that within two days a vessel named "Black Pearl" is about to enter the zone under surveillance with

3 `https://www.ihs.com`
4 `http://www.marinetraffic.com`
5 Number of registered ships on `http://www.fleetmon.com` (April 2013)

smuggling goods on board. This sets the operator to start carefully examining the vessels within the area of interest. The problem is that there can be multiple vessels with this name and the smuggling vessel might hide its identity by transmitting a wrong one.

Suppose that the operator examines the smuggling vessel, without, of course, knowing that it is the smuggling one. The vessel has the following true identity information:

$$mmsi = 123456789, name = \text{``}Black\,Pearl''$$

The vessel however transmits an AIS message with the following information:

$$ais : mmsi = 123456789, name = \text{``}Dutchman''$$

To verify this pretended identity, the operator retrieves additional information from IHS Fairplay (*ihs*) and MarineTraffic (*mt*):

$$\begin{aligned} ihs &: mmsi = 123456789, name = \text{``}Black\,Pearl'' \\ mt1 &: mmsi = 987654321, name = \text{``}Dutchman'' \\ mt2 &: mmsi = 123456789, name = \text{``}Black\,Pearl'' \end{aligned}$$

There is obviously a contradiction between the names reported by AIS and by IHS Fairplay, and there might be several possible interpretations, e.g. (*i*) the vessel might send out a wrong name, intentionally or due to an input error, (*ii*) the name in the IHS Fairplay record is wrong, or (*iii*) the MMSI sent out by the vessel is wrong and the IHS Fairplay record is about a different ship. On the other hand, the first MarineTraffic record is not that likely about the vessel under examination, since there may be multiple vessels named "Dutchman". The second MarineTraffic record confirms the IHS Fairplay information.

Given this information, the operator suspects that the vessel is trying to hide its identity by sending a wrong name, and thus the vessel might be involved in smuggling. Information from IHS Fairplay is usually very reliable, although uncertainty about correctness of that information always remains as well. To verify these hypotheses, the operator requires a patrolling boat to visually observe the vessel at distance. The information reported back is that the vessel has been repainted at sea, which gives the operator further support for the hypotheses that this vessel is trying to hide its identity and is smuggling.

Although this scenario is a simplified version of reality, it clearly illustrates the complexity of maritime SaS tasks and the realistic problem of reasoning about objects under surveillance using all available information at once.

8.3 A PCLP MODEL FOR MARITIME SAS TASKS

We here present the main concepts of the PCLP model, which we built for maritime SaS tasks, referred to as the *Probabilistic Maritime Safety and Security Model* (PMSM) hereafter. Since the capabilities of the model have to fit the intended environment and purpose for which it is employed, we note that the solutions presented here were mostly driven by the context of the METIS project. Alternative solutions may be needed in other contexts, even when modelling the same domain. We therefore focus on the main ideas of our approach to information fusion and give some examples of how we filled in details, with the main purpose of demonstrating the applicability and benefits of PCLP.

8.3.1 *Syntactic Sugar*

We use PCLP as modelling language, but add some syntactic sugar for defining distributions, to increase readability. First, there are some abbreviations we use to denote distributions used often:

$$\begin{aligned} \mathit{const}(C) &\equiv \{1.0 : C\} \\ \mathit{flip}(P) &\equiv \{P\colon \mathit{true}, 1 - P\colon \mathit{false}\} \\ \mathit{uniform}([V_1, \dots, V_n]) &\equiv \{1/n\colon V_1, \dots, 1/n\colon V_n\} \end{aligned}$$

We furthermore use sugar for mixtures of distributions:

$$\mathit{mix}(p_1\colon \mathit{Dist}_1, \dots, p_n\colon \mathit{Dist}_n),$$

where $p_1 + \cdots + p_n = 1.0$ and $\mathit{Dist}_1, \dots, \mathit{Dist}_n$ are distribution definitions themselves. This mixes the given distributions weighted by the associated probabilities.

Also we add syntax for conditional definitions to increase readability. As discussed, in an *imprecise* setting one cannot express conditional probabilities in PCLP, but we build a precise model here. Approximation of precise probabilities for models with conditional definitions by imprecise bounds is still possible with arbitrary precision, as shown in Chapter 6. Conditional definitions have the form:

$$\begin{aligned} &\mathbf{V} \sim \mathrm{Def}_1 \leftarrow \mathrm{Cond}_1 \\ &\mathbf{V} \sim \mathrm{Def}_2 \leftarrow \mathrm{Cond}_2 \\ &\dots \\ &\mathbf{V} \sim \mathrm{Def}_n \end{aligned}$$

Each definition Def_i is an arbitrary distribution definition and each condition has the same form as a *logic programming* (LP) rule's body. It therefore represents a conjunction of literals: $l_1, \dots, l_n$. The semantic of such definition is that $\mathbf{V}$ is defined by the first definition, of which the condition holds. The last definition is a default case, applied in case none of the conditions of the rules before applies. We provide sugar for the special conditional case hat a *random variable* equals another one:

$$\mathbf{Y} \sim \mathit{equals}(\mathbf{X}) \equiv \mathbf{Y} \sim \mathit{const}(V) \leftarrow \langle \mathbf{X} = V \rangle \text{ for all values } V \text{ in the range of } \mathbf{X}$$

8.3.2 *Modelling The Domain Entities and Their Properties*

Although the presented modelling approach applies in the same way to various kinds of objects, e.g. vessels, persons, companies, we restrict here to vessels in favour of a concise description of the model. A schematic representation of the domain concepts is given in Figure 8.1.

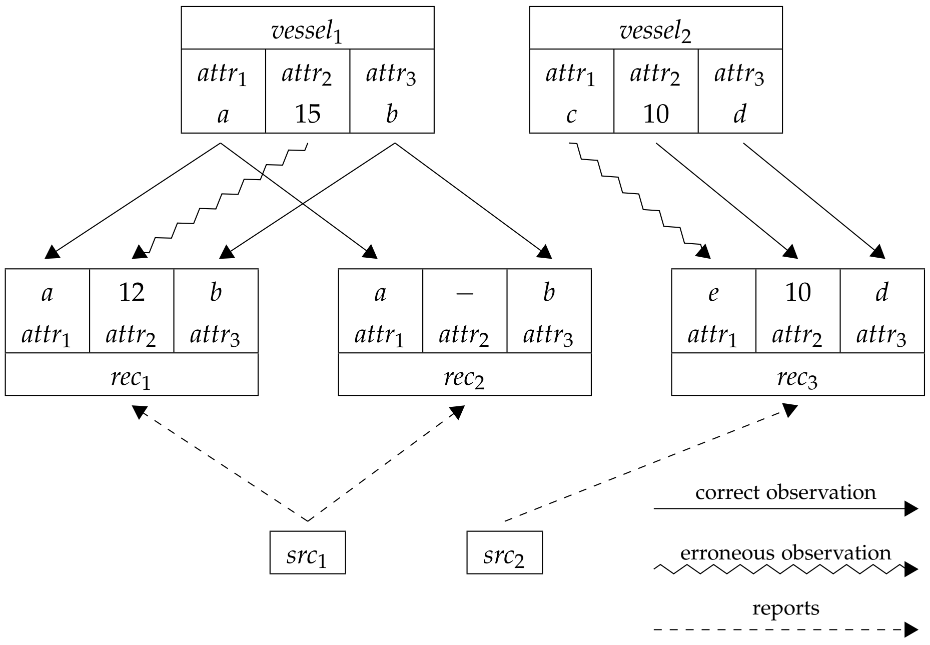


Figure 8.1: Relationships Between Vessels, Records and Sources

We assume a fixed set of vessels in the real world, represented by labels $vessel_1, \dots, vessel_n$, where we estimated n based on data, as described in Section 8.2. We model general knowledge about vessels as first-order rules, parametrised by such labels in PCLP. This allows us to model vessels' attributes and the observed information about them in the same uniform way.

Each of the vessels under surveillance is described by the same attributes $attr_1, \dots, attr_m$. Attributes can be low-level information like names or high-level information concerning intentions, e.g. indicating that a ship is smuggling. Although the two attribute levels are semantically different, in this model we do not make a distinction between them and represent them in the same way. In particular, the value of the attribute *Attr* of vessel *Vessel* is defined as a parametrised random variable:

$$\mathbf{Attr}(Vessel, Attr)$$

For the label of attributes we use names from the domain's data model, e.g. *category* and *name*. An important common property is that all attributes behave like functions, which means each vessel attribute can only have one value at a time.

A *source* reports a number of records $rec_1 \ldots, rec_k$ and the fact that a record *Rec* is reported by a source *Src* is represented by:

$$source(Rec, Src)$$

We use a *predicate* instead of a random variable here, as we assume there is no uncertainty about the source we get a record from. Furthermore, we assume that each record is always related to a single vessel. For example, in Figure 8.1 rec_1 and rec_2 are about $vessel_1$ and rec_3 is about $vessel_2$. Attribute values in records are observations of the vessels' actual attribute values, but records do not have to provide values for all attributes. The record rec_2 for instance does not provide a value for attribute $attr_2$. Records can report erroneous attribute values, which means that the value is not equal to the observed vessel's actual one. For instance, rec_1 reports that the value of $attr_2$ for $vessel_1$ is 12, although the attribute's actual value for that vessel is 15. Again a parametrised random variable definition represents the values of attributes in records:

$$\mathbf{RecAttr}(Rec, Attr)$$

We use the convention that the label of each record consists of the source name with an attached numeric identifier, e.g. a record with a label *intel1* is provided by an intelligence report. Other examples are *ais1*, *fairplay1* and *marinetraffic1*.

The fact that values are missing (not reported) can itself be useful input information for the probabilistic model. Representing missing information is in itself a complex modelling task, where different types of missingness need to be considered, e.g. whether there was no attempt to retrieve information or whether there was an attempt but the source did not provide it. While this is a valuable research direction, for the purpose of this work, we focus only the values available (reported) in records.

8.3.3 *Modelling Vessels' Attributes*

The basis of the PMSM is a sub-model of the properties and behaviours of vessels, represented by their attributes. This sub-model does not dependent on observed information, as we follow an approach of causal modelling: the observations are effects of the true vessel attributes. The relations within the sub-model are essential for interpreting observations because many attributes cannot be observed directly but only through observations of other attributes. For instance, the observation that a vessel is a tanker also reveals something about its tonnage. In addition, it is often only possible to draw conclusions about intention-related attributes based on observations of low-level attributes.

Concretely, the sub-model of vessel attributes takes the form of a probability distribution on the attributes. Such distribution can be modelled by conditional PCLP definitions of the random variables $\mathbf{Attr}(Vessel, Attr)$, for different groundings of *Attr*. We always quantify over the variable *Vessel*, as we build a general model valid for all vessels. The distributions may dependent on other random variables with label *Attr*, but not on records and sources. Thanks to the expressive power of PCLP we can use any kind of range, discrete or continuous, to model the different attributes and specify conditions in a natural way, as illustrated by the following example. We do not consider machine learning in this thesis, but obviously distributions learned from data about physical attributes, as the tonnage depending on the category, can be incorporated here.

Example 8.1

The prior probability that a ship is smuggling with a probability of 1% can be represented as:

$$\mathbf{Attr}(\mathit{Vessel}, \mathit{smuggling}) \sim \mathit{flip}(0.01)$$

The probability that a vessel is hiding its identity is much higher in case it is smuggling. This can be expressed by:

$$\begin{aligned}&\mathbf{Attr}(\mathit{Vessel}, \mathit{hidesIdentity}) \sim \mathit{flip}(0.7\ \ ) \leftarrow \langle \mathbf{Attr}(\mathit{Vessel}, \mathit{smuggling}) = \mathit{true}\rangle\\ &\mathbf{Attr}(\mathit{Vessel}, \mathit{hidesIdentity}) \sim \mathit{flip}(0.01)\end{aligned}$$

We model gross tonnage as a continuous variable with two different distributions, one for tankers and one for all other vessels:

$$\begin{aligned}&\mathbf{Attr}(\mathit{Vessel}, \mathit{grossTonnage}) \sim \mathit{gamma}(0.72, 36\,000.0)\\ &\qquad\qquad\qquad\qquad\qquad\qquad \leftarrow \langle \mathbf{Attr}(\mathit{Vessel}, \mathit{category}) = \mathit{tanker}\rangle\\ &\mathbf{Attr}(\mathit{Vessel}, \mathit{grossTonnage}) \sim \mathit{gamma}(0.76, \ \ 3\,300.0)\end{aligned}$$

Even though we aim to express all knowledge required in a natural way, practical model construction sometimes requires certain workarounds to obtain a workable model. An advantage of an expressive modelling languages as PCLP is however that such workarounds can be expressed concisely in the language itself.

An example of this is that we need a special construct for attributes with huge cardinality. Examples for this are the IMO number and the name of vessels. Note that, although the IMO is a number, we should not model it as integer variable with a known distribution. We should instead treat is as 10 000 000 distinct labels, without any order. So we model it in principle as a finite distribution over discrete constants. This is however problematic, as the number of states is too high for the inference algorithm we use. For the name attribute an additional complication is the fact that we do not know all possible names in advance.

We handle this complexity by not reasoning about all vessels at the same time, but about each vessel separately. We refer to the single ship we currently reason about as *vessel of interest* (VoI) and denote it with the special constant *voi*. This is also necessary to reduce the amount of information we reason about, as we also cannot reason about all available pieces of information at the same time.

By reasoning about a single vessel at a time we can restrict to a small subset of the information available, which is potentially relevant for the current VoI. The method to represent distributions of attributes with huge cardinality is based on the idea that most values are indistinguishable with respect to the VoI (e.g. all IMO numbers that do not occur in the information reported) and we can summarise them with a special constant, we call *other*.

To model such distributions we make use of a special predicate, which is used to generate the actual distribution:

$$dynamicRange(Attr, FixedPart, TotalNr)$$

The starting point here is a uniform distribution, where each element has a probability of $1/TotalNr$, where *TotalNr* is the total number of distinct values for this attribute, and *other* gets the rest of the probability mass. For some attributes we however have to consider some additional special values and mix another distribution to the uniform one, which is given as *FixedPart*. This parameter has to be a probability-distribution pair or *none* in case no such distribution is required.

Thanks to the first-order nature of PCLP, the actual distribution can automatically be generated from such specifications, using general rules. The rule for the case that there is no *FixedPart* distribution specified is:

$$\begin{aligned}\mathbf{Attr}(Vessel, Attr) \sim mix(\quad & NrObs/NrTotal\colon uniform(Values),\\ & 1 - NrObs/NrTotal\colon const(other)\,)\\ \leftarrow\ & dynamicRange(Attr, none, NrTotal),\\ & observedValues(Attr, Values),\\ & length(Values, NrObs)\end{aligned}$$

We assume that the predicate *observedValues* is defined such that it indicates the observed values for the attribute, i.e. the values that have to be treated in a distinct way. Each of such value gets a probability mass of $1/NrTotal$ and the rest of the probability mass is assigned to the special value *other*. This is achieved by mixing a uniform distribution over all observed values with the distribution with constant value *other*, using for the latter the weight $NrObs/NrTotal$, where *NrObs* is the total number of such observed values. Analogously, a rule can be defined for the case an additional distribution is mixed in, which is omitted here.

Example 8.2

The distribution for vessels' names is specified as follows:

$$dynamicRange(name, none, 380\,000 \cdot 0.8)$$

This defines a uniform distribution over $380\,000 \cdot 0.8 = 304\,000$ distinct names (see Section 8.2). For instance, in case the names *"Dutchman"* and *"Black Pearl"* are observed, this yields the following distribution:

$$\{1/304\,000\colon \text{"Dutchman"}, 1/304\,000\colon \text{"Black Pearl"}, 303\,998/304\,000\colon other\}$$

Half of the vessels do not have an IMO number. We represent this case with the constant *noIMO*. The distribution is therefore specified as:

$$dynamicRange(imo, 0.5\colon const(noIMO), 380\,000 \cdot 0.5)$$

In case we observe the IMOs 1234567 and 7654321 this yields:

$$\{\ 0.5 \cdot 1/190\,000\colon 1234567, 0.5 \cdot 1/190\,000\colon 7654321,\\ 0.5 \cdot 189\,998/190\,000\colon other, 0.5\colon noIMO\ \}$$

8.3.4 *Relating True and Observed Vessel Information*

As discussed in the introduction one of the main challenges for automated reasoning in the maritime SaS tasks is the uncertainty in the observed vessel information. The probabilistic nature of the model is imposed due to the uncertainty in the relationships between the vessels' actual attribute values and the values reported by the records. This uncertainty has two main sources. First, the reported information can be erroneous or not. The binary random variables **Error**(*Rec*, *Attr*) indicates whether the attribute *Attr*, reported by the information record *Rec*, is erroneous. Second, it is unknown to which actual vessel the records belong. This aspect is related to the problem of *entity resolution* [68, 104] and more complex than the first one. About which vessel a record provides information, is represented by the random variables **About**(*Rec*).

The goal of the PMSM is to predict the VoI's true attributes using potentially relevant observations. To achieve this goal, we use two rules, one for erroneous and one for non-erroneous information, to relate the true and observed vessel attributes:

$$\begin{aligned}
\mathbf{RecAttr}(Rec, Attr) \sim\ & equals(\mathbf{attr}(Vessel, Attr)) \leftarrow \\
& \langle \mathbf{Error}(Rec, Attr) = false\rangle, \langle \mathbf{About}(Rec) = Vessel\rangle \\
\mathbf{RecAttr}(Rec, Attr) \sim\ & Dist \leftarrow \\
& \langle \mathbf{Error}(Rec, Attr) = true\rangle, \langle \mathbf{About}(Rec) = Vessel\rangle, \\
& errorDist(Vessel, Attr, Dist)
\end{aligned}$$

In case the information is correct, the value reported equals the actual attribute's value of the vessel that the record is about, as expressed by the first rule. For the opposite case, it is necessary to determine a distribution, modelling the characteristics of errors, which is done by *errorDist*.

The issue of erroneous information is strongly related to classical information fusion, where measures of the same quantity of interest are used to obtain a prediction of that magnitude, based on a model of the sensors' error characteristics. However, PCLP allows and the domain requires to consider relations between intentions and the probability of erroneous information, whereas classical information fusion often only considers non-intentional errors due to noisy measurements. While in practice a simple model, e.g. for discrete values a uniform distribution leaving out the correct value, has shown to be sufficient to model erroneous values, it is essential to relate the probability of erroneous information to various factors. We therefore do not give details of the rules defining *errorDist* here and restrict to illustrating how the error rate can be modelled by the following example.

Example 8.3

Suppose we judge the error rate of all records from the IHS Fairplay database to be 2% This can be expressed by:

$$\mathbf{Error}(\mathit{Rec}, \mathit{Attr}) \sim \mathit{flip}(0.02) \leftarrow \mathit{source}(\mathit{Rec}, \mathit{fairplay})$$

AIS messages are handled in a special way. First, we know for sure that they are about the VoI, though of course all information provided may be completely wrong. In case the VoI is hiding its identity it will certainly send out a wrong name, since the name is always mentioned in news articles and other reports. There is also a high chance that a wrong MMSI and IMO is sent. There are not many reasons to hide other attributes like the category, since they cannot be used to identify a vessel. Finally, the last rule represents the case that there is no attempt to hide a vessel's identity. Still, information can be erroneous due to unintentional mistakes with a chance of 5%:

$$\begin{array}{lll}
\mathbf{Error}(\mathit{ais}, \mathit{name}) & \sim \mathit{const}(\mathit{true}) & \leftarrow \langle \mathbf{Attr}(\mathit{voi}, \mathit{hidesIdentity}) = \mathit{true} \rangle \\
\mathbf{Error}(\mathit{ais}, \mathit{mmsi}) & \sim \mathit{flip}(0.6) & \leftarrow \langle \mathbf{Attr}(\mathit{voi}, \mathit{hidesIdentity}) = \mathit{true} \rangle \\
\mathbf{Error}(\mathit{ais}, \mathit{imo}) & \sim \mathit{flip}(0.7) & \leftarrow \langle \mathbf{Attr}(\mathit{voi}, \mathit{hidesIdentity}) = \mathit{true} \rangle \\
\mathbf{Error}(\mathit{ais}, \_) & \sim \mathit{flip}(0.1) & \leftarrow \langle \mathbf{Attr}(\mathit{voi}, \mathit{hidesIdentity}) = \mathit{true} \rangle \\
\mathbf{Error}(\mathit{ais}, \_) & \sim \mathit{flip}(0.05) & \leftarrow \langle \mathbf{Attr}(\mathit{voi}, \mathit{hidesIdentity}) = \mathit{false} \rangle
\end{array}$$

This demonstrates that PCLP allows to express complex dependencies in an intuitive manner.

Modelling the relation between true and observed vessel information is challenging and the solution presented here demonstrates the capabilities of PCLP to allow for an elegant representation of complex relationships with inherent uncertainty. The solution presented, based on the random variables **About**(*Rec*), is a unique result of the work presented in this chapter.

The random variable **About**(*Rec*) can straightforwardly be defined for information records we certainly know are about the VoI. There always has to be such information, as otherwise it is impossible to draw any specific conclusions about a vessel. An example is that we know a vessel sends out an AIS message and have some radar information about its size. For all other records, for instance from databases, we use a very small prior probability that it is about the VoI, which is one divided by the estimated number of vessels in the domain.

The posterior however can be much higher, because matching attribute values increase the chance that the record is about a vessel.

If we would reason about all vessels at the same time, we could use the same prior for all of them. As this is not practical, we use a similar workaround as for attributes and introduce special labels that represent sets of indistinguishable vessels. This can again be realised by concise rules in the model itself, thanks to the expressive power of PCLP. We however omit the details here.

8.4 model application and evaluation

To get insights about the applicability and performance of the proposed PMSM, we applied it to the scenario from Section 8.2.3 and conducted experiments with simulated and real-world vessel data. Finally, we give some results of how the model performs embedded in a complex prototype system for maritime SaS tasks, which we tested with real data.

8.4.1 *Application to the Maritime Scenario*

We apply our model to the maritime scenario presented earlier (Section 8.2.3). The observations are represented by the evidence:

$$\begin{aligned}
&\mathbf{RecAttr}(ais, mmsi) = 123456789,\\
&\mathbf{RecAttr}(ais, name) = \text{“}Dutchman\text{''},\\
&\mathbf{RecAttr}(ihs, mmsi) = 123456789,\\
&\ldots
\end{aligned}$$

We demonstrate how the amount of available evidence changes the reasoning result by using three evidence sets: only the AIS and IHS Fairplay information (*ihs*), the AIS and IHS Fairplay information with added information from MarineTraffic (*ihs+mt*) and finally we add the visual observation (*vis*) that the vessel has been repainted (*ihs+mt+vis*):

$$\mathbf{RecAttr}(visualsign, repainted) = true$$

Table 8.1 reports the query probabilities. The values of 0.000 and 1.000 indicate rounded very small and large probabilities. We abbreviate queries of the form $\mathbf{Attr}(voi, Attr) = Value$ with $Attr = Value$.

The intuitive line of reasoning for the scenario is that the vessel seems to send the wrong name “Dutchman” in order to hide its identity, which is confirmed by the model if sufficient information is available. With IHS Fairplay information only, the probability that the real name is “Black Pearl” is only about 85%, though we trust IHS Fairplay much more than AIS information. This is because it is not certain whether the record is about the ship of interest, since the matching MMSI is the only link between the ship and the record. With an increasing amount of information available, the probabilities change to the expected result. Another

Query	*ihs*	*ihs+mt*	*ihs+mt+vis*
$mmsi = 123456789$	1.000	0.964	0.992
$mmsi = 987654321$	0.000	0.035	0.007
$mmsi = other$	0.000	0.001	0.001
$name =$ *"Dutchman"*	0.146	0.037	0.008
$name =$ *"Black Pearl"*	0.845	0.963	0.992
$name = other$	0.009	0.000	0.000
$smuggling = true$	0.037	0.042	0.335
$hides\_identity = true$	0.083	0.094	0.806

Table 8.1: Results for the Example Scenario

interesting observation is that information about the behaviour influence the probability distributions of the low-level attributes and vice versa.

8.4.2 *Quantitative Evaluation Using Simulated Data*

We quantitatively evaluate the PMSM by showing that it can correct errors in simulated AIS data by using additional information. We simulate 1 000 vessels according to the prior distribution we estimated analysing the data. For each source, we generate a record corresponding to a vessel and randomly introduced errors with a fixed error rate for each attribute; thus, we create erroneous observations (cf. Figure 8.1). The record is then saved to a database, without any link between the object and the record. Similarly, we generate AIS messages for each vessel with randomly generated errors. The errors are generated in such a way that the chance to accidentally report another vessel attribute's value matches the estimations. In the experiments we vary the error rate of AIS messages from 0.0 to 1.0 and denote it with ϵ_A^{AIS}. An error rate of 0.0 means that values reported in AIS messages are correct for all attributes, whereas an error rate of 1.0 means all values are incorrect.

We evaluate the error of the prediction, which is the most likely value according to the result of the PMSM with the AIS message and additional information as input. The additional information used by the PMSM model is retrieved from the database using the values of the MMSI and IMO in the AIS message. Due to the errors there may be multiple records that include the same MMSI or IMO or no record including a sent MMSI or IMO at all. Clearly, because of this there was a varying number of relevant records per vessel.

We measure the performance of the predictions in terms of the error per attribute A:

$$\epsilon_A^{\text{PMSM}} = \frac{\text{incorrectly predicted values for } A}{\text{total number of vessels}}$$

In the experiment, we explore whether sources with high error rates are still useful to correct errors and whether the error reduction increases with the number of distinct noisy sources. We choose an error rate for the sources of 20% ($\epsilon = 0.2$), which could be considered as very noisy in practice, and vary the number of sources from 1 to 3. As baseline we use a single source, providing perfect information ($\epsilon = 0.0$). This case is still not trivial, since, due to errors in the AIS data, the record corresponding to the current ship of interest might not be found and instead records about other vessels are used as evidence.

Since we obtain similar results for all attributes, for clarity we discuss only the ones for the vessel's name; see Figure 8.2, where we plot the error of the input AIS data (ϵ_A^{AIS}) against the error of the prediction ϵ_A^{PMSM}. A first observation is that for moderate error rates of up to 20% for the AIS messages, a single noisy source can not or hardly correct for errors. Adding a second and a third source decreases the error rate of the predicted attributes. In the latter case, the errors are reduced nearly as much as using a source without errors. With an increasing error rate in the AIS message, the difference between the predicted error rates decreases. However, those high error rates are very unlikely to occur in practice.

We can draw the following conclusions. First, our approach is capable of correcting errors in AIS data to a certain extent. Second, for AIS error rates of up to 20%, which are not expected to be higher in practice, most errors could be corrected given reliable additional information. Finally, we confirmed that our model can make good use of very noisy information sources. The low quality of information can be compensated by the amount of information used. This result is similar to typical results for ordinary multi-sensors fusion. It however confirms that our method is in a similar way capable of correcting errors by relating information from external databases.

8.4.3 *The Model Embedded in a Real Maritime System*

Within the METIS project, a proof of concept system was developed, tackling all aspects of automating the maritime SaS task, including data collection, dynamic reconfiguration and visualisation, next to reasoning. The PMSM is however an integral part of the entire system, so the prototype illustrates the applicability of the PMSM in a real-world setting. The prototype system was deployed as a plugin for Thales's command-and-control industrial platform *Tacticos*. Figure 8.3 depicts the main working principles of the prototype. A detailed description of the system's general architecture is furthermore given in [65].

Initially, the operator or the system selects a single vessel of interest, whose AIS identity information, as given in the scenario, is used to retrieve data from additional sources. All the information retrieved is then aligned to a common information model[6]. The aligned information serves as an input to the PMSM, of which the results are then visualised for the operator as shown in Figure 8.4. Dy-

6 `https://www.niem.gov/communities/maritime`

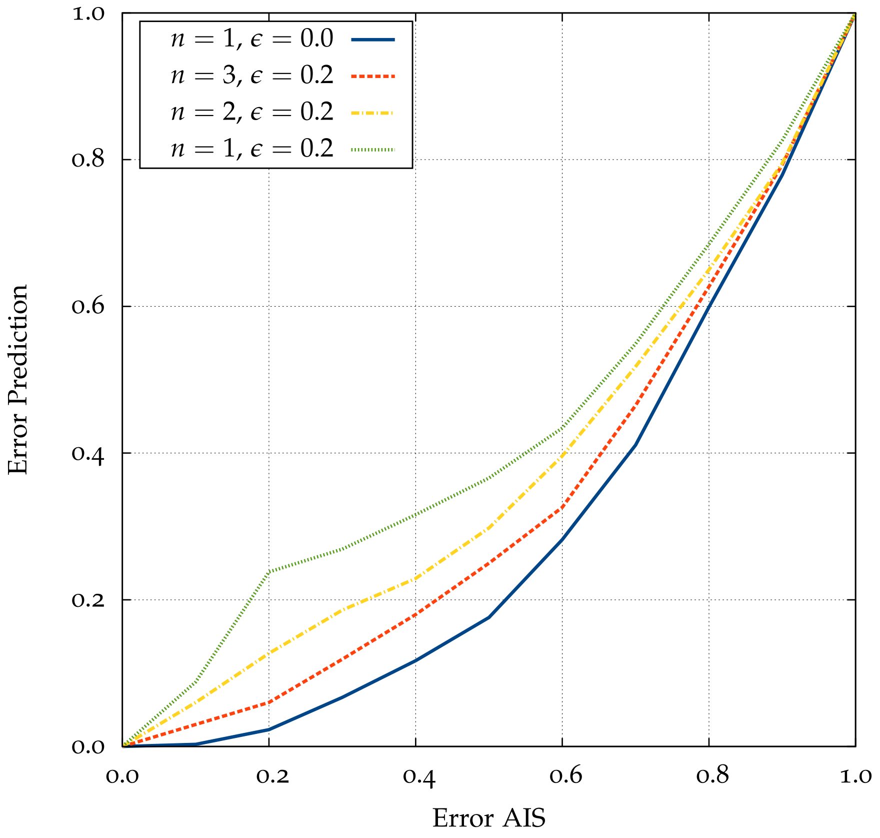


Figure 8.2: Prediction Error of the PMSM for Names

namic reconfiguration can decide to do the process over again, using additional information sources.

A 24/7 system, driven by live AIS data distributed by *AIS Hub*[7] and continuously (re)deployed with the latest state-of-the-art METIS technologies, started at the beginning of 2014 to continuously monitor all ship activities within the Dutch exclusive economic zone. To give an insight in the system's performance we give a short summary of its results. For the period May 18–21 the METIS system has monitored around 9 400 ships. Information about vessels from sources such as AIS, IHS Fairplay and the *Press Association*[8] was constantly collected and fused to reason about vessels' identities and intents such as reckless behaviour, smuggling and environmental hazard. The process of data collection and reasoning took on average a few seconds per vessel, affirming the real-time operation

7 http://www.aishub.net
8 https://www.pressassociation.com

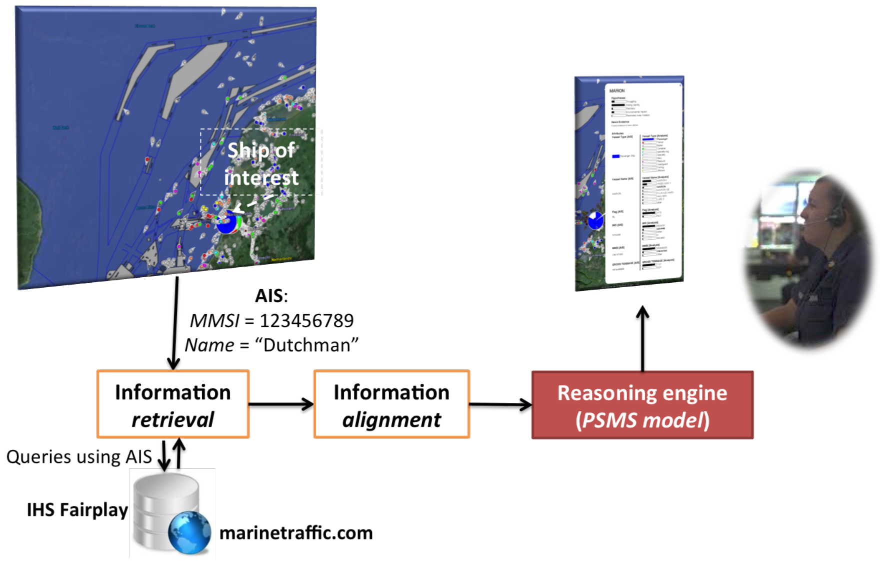


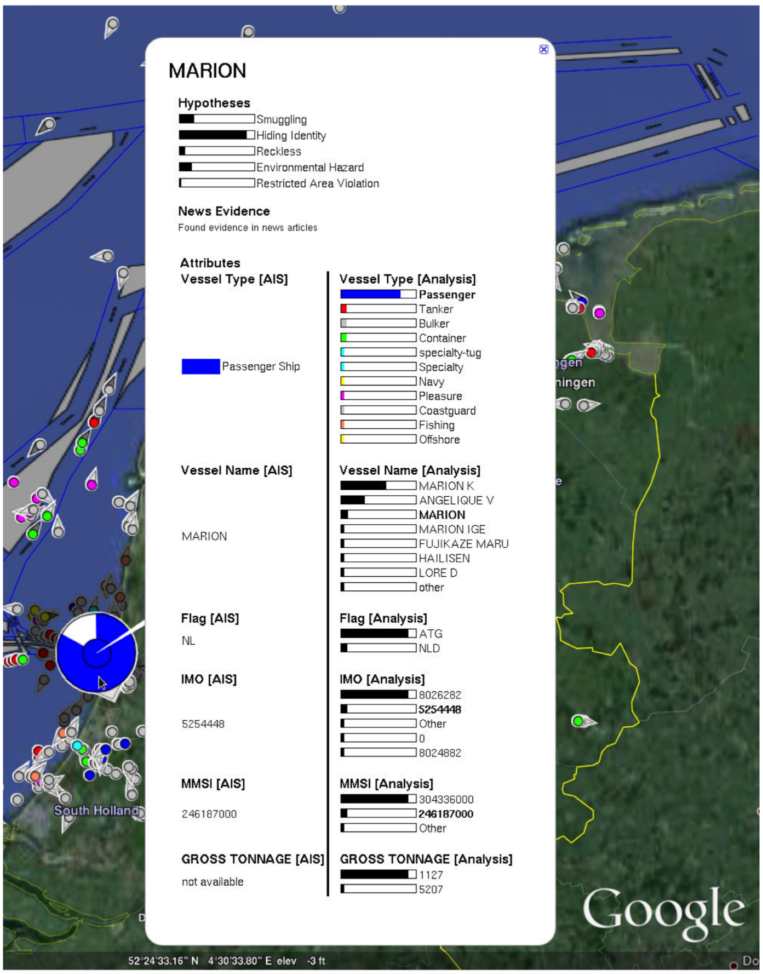
Figure 8.3: Operational Scheme of the METIS Prototype System

of the METIS system. The system made use of an alert rate, which is based on the probability of relevant hypothesis. For 99% of the monitored ships, the system reported an alert rate of less than 10% for any of the intents. For the remaining ships, 0.94% had an alert rate of up to 41% mostly due to a high (prior) chance for reckless behaviour. Only for two ships, the alert rates were 78% and 92% due to news evidence about reckless behaviour. These results demonstrate the potential of the system to filter out a significant part of the ships entering a monitored area.

8.5 RELATED WORK

The distinction we make between low-level attributes, such as physical and identity attributes, and high-level attributes, related to intentions, is also commonly made in the context of information fusion. For instance, Waltz and Llinas discuss the distinction between *low-level processing* and *high-level processing* [155]. As discussed, the problem of low-level processing for kinematic properties is well studied [52, 59], but by its nature involves less kinds of uncertainty.

Other work exclusively focuses on the high-level processing task, using results of low-level processing as input, assuming that such inputs are generated from sensors under own control, of which the error characteristics are independent of high-level intentions. Such low-level information never reveals the identity of objects. Early works only aimed at inferring a possible single threat from low-level data, e.g. oil spills [78]. Later work aimes at providing a more complete picture and is as our work also based on first-order probabilistic languages: the work in [21, 28] is based on *multi-entity Bayesian networks* [84] and the work



MARION

Hypotheses

- Smuggling
- Hiding Identity
- Reckless
- Environmental Hazard
- Restricted Area Violation

News Evidence

Found evidence in news articles

Attributes

Vessel Type [AIS]	Vessel Type [Analysis]
Passenger Ship	Passenger Tanker Bulker Container specialty-tug Specialty Navy Pleasure Coastguard Fishing Offshore
Vessel Name [AIS]	**Vessel Name [Analysis]**
MARION	MARION K ANGELIQUE V MARION MARION IGE FUJIKAZE MARU HAILISEN LORE D other
Flag [AIS]	**Flag [Analysis]**
NL	ATG NLD
IMO [AIS]	**IMO [Analysis]**
5254448	8026282 5254448 Other 0 8024882
MMSI [AIS]	**MMSI [Analysis]**
246187000	304336000 246187000 Other
GROSS TONNAGE [AIS]	**GROSS TONNAGE [Analysis]**
not available	1127 5207

Figure 8.4: Screenshot of the METIS Prototype System

in [144] on *Markov logic networks* (MLNs) [44]. An approach taking into account the behaviour of vessels over a period of time is presented in [49] and applies *Dynamic Bayesian Networks* [108]. An example of similar work outside of the maritime domain is [147].

There are also other systems, aiming at an integration of all aspect of SaS support systems, as we did in the METIS project. One example of such work that deals with the interpretation of uncertain and missing information is presented in [109]. The reasoning engine is based on Dynamic Bayesian Networkss, while a situation recovery component provides the final decision-making. The disadvantage of the employed tool, however, is its propositional nature that does not allow dealing with a varying number of entities as existing in the maritime domain. In [23], the authors report about the *GeMASS* system that uses genetic algorithms to discover knowledge from historical AIS and local port management data about common and exceptional events of ships' kinematic behaviour.

8.6 conclusions & future work

We have developed a model for decision support for maritime SaS tasks based on PCLP. In contrast to previous work we deal with the challenging problem of reasoning about vessels' identities, which is essential for utilising information from external sources. This kind of reasoning comes with the additional uncertainty that error characteristics of information sources, not under own control, may depend on vessels' intentions, which implies complex relations between vessels and pieces of information.

PCLP has shown to be a very powerful tool to deal with such complex relations between dynamic amounts of information and a dynamic number of vessels. The representation of domain knowledge makes exposition of this knowledge possible in several ways. This is very important, because an important lesson learned in the course of the METIS project, is that there is no single best way of doing things, but the capabilities of the model and used input information have to fit the intended environment and purpose for which it is employed. Often more simple sub-models of certain aspects of the domain can provide results that are good enough, while a too complex model hinders maintainability and increases inference time. Also the model is part of a complex support system, for which interfacing with human experts is crucial.

The model is capable of fusing information by relating information from heterogeneous sources to objects under surveillance. We verified that the model can correct errors in the information transmitted by vessels using information provided by external sources, which are unreliable as well. The number of sources increase the reliability of the model's prediction. We have finally shown that our solution is applicable in the context of a realistic support system prototype.

We believe that the general structure of the model, that is the relation of objects of interest and reported information, to be widely applicable. The pattern of having a complex interplay between properties of a dynamic number of entities of interest and uncertain information about such entities, originating from sources with different characteristics, occurs in many potential application areas.

Some technical improvements are possible in the future. The workaround to deal with large ranges, using the special value *other*, works similar to lifted inference methods. The method has therefore to be seen as a workaround for a weakness shared by all non-lifted inference algorithms. So improved inference algorithms may automate this workaround, making an even cleaner and more declarative model possible.

9 DISCUSSION AND CONCLUSIONS

This thesis covers the whole spectrum of work around the development of *probabilistic logics*: from language design, to an investigation of the theoretical properties of the designed languages, the development of efficient probabilistic reasoning algorithms and their implementation, and finally the development of an application. We conclude the thesis by discussing the achievements and lessons learned from the work. Furthermore, some future research opportunities are proposed.

9.1 ACHIEVEMENTS

The different aspects of what has been achieved by the work presented in this thesis is discussed in the following.

9.1.1 *Theoretical Contributions*

We propose *Probabilistic Constraint Logic Programming* (PCLP), a language which possesses a novel balance between expressiveness, consistency properties and *inference* complexity. The language supports reasoning about *imprecise* distributions, where the distributions are guaranteed to remain consistent. In addition, inference is made possible with a generalised version of a state-of-the-art inference algorithm based on *weighted model counting* (WMC), inheriting many of the advantages of this method. At the same time, only a small price has to be paid in terms of inference complexity. The work on *imprecise probabilistic Horn clause logic* (IPHL) offers another alternative by limiting the expressiveness in terms of the problem's structure, eliminating the increase in terms of parametrised complexity.

The results illustrate the complex interplay between expressiveness, suitability for knowledge representation and inference complexity for probabilistic logic languages. On the one hand, it is possible to *soundly* reason about problems with underspecified probabilistic knowledge and problems for which exactly computing probabilities is impossible by only considering an imprecise approximation. On the other hand, reasoning about imprecise knowledge in general increases inference complexity and only carefully chosen properties of languages, as those of PCLP and IPHL, can avoid that increase.

9.1.2 *Implementation & Experiments*

We developed a reference implementation of the generalised WMC algorithm, which we used to illustrate that the overhead of computing bounds instead of point probabilities is negligible. It was also shown that additional exploitation of structure in problems can reduce the inference time essentially, similarly to using optimisations in the context of exact inference. Furthermore, we implemented an algorithm based on the ideas of generalised WMC, aiming at efficiently computing bounds for precise, but hybrid, problems. An additional aspect, tackled by this work, is how to discretise the continuous *random variables* to achieve quick convergence of the probability bounds for the entire problem.

This illustrates that the theoretical insights can be transferred to practical implementations. Those implementations are only possible by incorporating insights from existing algorithms, in this case WMC algorithms, and also by reusing implementations, in this case *satisfiability modulo theories* (SMT) solvers.

9.1.3 *Successful Applications*

The language was successfully deployed for building a model of safety and security tasks in the maritime domain. This shows that PCLP is suitable as a knowledge representation language in domains for which data is scarce. Furthermore, the properties of PCLP offered the possibility of integration into a complex prototype system for maritime situational awareness, as the logic-based approach provides useful information about the internal state of reasoning.

This also reveals the practical benefits of using methods based on logic. The properties provided by such methods are essential for incorporating knowledge and integrating reasoning capabilities into a complex system that includes human operators.

9.2 LIMITATIONS

A limitation of PCLP is that it is based on an extended version of the *distribution semantics*, inheriting some of the typical limitations of languages that employ such semantics. The language is limited to closed universe models, while the ideas behind PCLP could probably also be used for open universe models, as for example realised by *BLOG* [102]. In general, the continuous distributions that can be approximated are restricted to non-parametric ones. For instance, one cannot use the value of a continuous random variable to determine the parameter of another one.

9.3 FURTHER RESEARCH

In the future the most important aspect to make PCLP practically useful is to supply the system with efficient inference mechanisms. This mainly means incorporating more results from recent research on precise inference. Concretely, this would imply using recent knowledge compilation techniques and to incorporate results from research about lifted inference. Inference for IPHL could benefit similarly. A possible remedy when continuous, approximate inference by discretisation fails, might be to combine this way of inference with sampling based approaches, to get the best of both worlds. The PCLP language could also be made more powerful by allowing parametric continuous distributions and open universe models. Also computing more than probabilities, for instance expectation values or best decisions, is a possible direction for future research. Finally, it is important to find more application domains for PCLP and IPHL to prove their practical applicability.

A PROOFS

Proof of Lemma 4.1. We show that the equation $P(q) = P_{\mathbf{V}}\big(\mathrm{SE}(q)\big)$ is equivalent to Definition 4.4 ($P(q) \overset{\text{def}}{=} P_{\mathbf{T}}\big(\{(\omega_{\mathbf{V}}, \omega_{\mathbf{L}}) \in \Omega_{\mathbf{T}} \mid \omega_{\mathbf{L}} \models q\}\big)$):

$$\begin{aligned} P(q) &= P_{\mathbf{T}}\big(\{(\omega_{\mathbf{V}}, \omega_{\mathbf{L}}) \in \Omega_{\mathbf{T}} \mid \omega_{\mathbf{L}} \models q\}\big) \\ &= P_{\mathbf{V}}\big(\{\omega_{\mathbf{V}} \mid (\omega_{\mathbf{V}}, \omega_{\mathbf{L}}) \in \Omega_{\mathbf{T}},\, M_{\mathbf{L}}(\omega_{\mathbf{V}}) \models \omega_{\mathbf{L}},\, \omega_{\mathbf{L}} \models q\}\big) \quad \text{(Definition 4.3)} \end{aligned}$$

The condition $M_{\mathbf{L}}(\omega_{\mathbf{V}}) \models \omega_{\mathbf{L}}, \omega_{\mathbf{L}} \models q$ is equivalent to $M_{\mathbf{L}}(\omega_{\mathbf{V}}) \models q$, since $M_{\mathbf{L}}(\omega_{\mathbf{V}})$ uniquely determines $\omega_{\mathbf{L}}$. Therefore:

$$\begin{aligned} P(q) &= P_{\mathbf{V}}\big(\{\omega_{\mathbf{V}} \mid (\omega_{\mathbf{V}}, \omega_{\mathbf{L}}) \in \Omega_{\mathbf{T}},\, M_{\mathbf{L}}(\omega_{\mathbf{V}}) \models q\}\big) \\ &= P_{\mathbf{V}}\big(\{\omega_{\mathbf{V}} \in \Omega_{\mathbf{V}} \mid M_{\mathbf{L}}(\omega_{\mathbf{V}}) \models q\}\big) \\ &= P_{\mathbf{V}}\big(\mathrm{SE}(q)\big) \qquad \text{(Definition 4.5)} \end{aligned}$$

□

Proof of Lemma 4.2. By applying Definitions 4.5 and 4.1 we have to prove that:

$$\{\omega_{\mathbf{V}} \in \Omega_{\mathbf{V}} \mid M_{\mathbf{L}}(\omega_{\mathbf{V}}) \models q\} = \Big\{\omega_{\mathbf{V}} \in \Omega_{\mathbf{V}} \mid \bigvee_{\substack{M_{\mathbf{L}}[\Phi] \models q \\ \Phi \subseteq \mathbf{Constr}}} \bigwedge_{\varphi \in \Phi} \varphi(\omega_{\mathbf{V}})\Big\}$$

($\subseteq$) Assume we have an $\omega_{\mathbf{V}}$, which is element of the left-hand set, i.e. for which $M_{\mathbf{L}}(\omega_{\mathbf{V}}) \models q$. We choose Φ such that $M_{\mathbf{L}}[\Phi] = M_{\mathbf{L}}(\omega_{\mathbf{V}})$. Since for $\omega_{\mathbf{V}}$ all constraints in Φ are satisfiable, $\bigwedge_{\varphi \in \Phi} \varphi(\omega_{\mathbf{V}})$ holds and therefore $\omega_{\mathbf{V}}$ is element of the right-hand set as well.
($\supseteq$) Assume we have an $\omega_{\mathbf{V}}$, which is element of the right-hand set, i.e. for which there is a Φ, such that $M_{\mathbf{L}}[\Phi] \models q$ and $\bigwedge_{\varphi \in \Phi} \varphi(\omega_{\mathbf{V}})$. Because of the latter, we know that $M_{\mathbf{L}}(\omega_{\mathbf{V}})$ includes $\langle\varphi\rangle$ for all constraints $\varphi \in \Phi$. From $M_{\mathbf{L}}[\Phi] \models q$ we can therefore conclude $M_{\mathbf{L}}(\omega_{\mathbf{V}}) \models q$, due to monotonicity of first-order logic. This means $\omega_{\mathbf{V}}$ is element of the left-hand set, too. □

Proof of Proposition 4.1. Assuming the three conditions hold, we prove that it is possible to compute the probability of an arbitrary query q as analytic expression. We show that the *solution event* of q is finite-dimensional, which means it has a finite representation, and can be computed in finite time. The probability of q can then be computed according to Lemma 4.1 assuming the *computable-measure condition*.

The solution event lemma (Lemma 4.2) makes use of subsets of the constraint language and derives unique models from that. The *finite-relevant-constraints condition* means that, in order to find all Φ for which $M_{\mathbf{L}}[\Phi] \models q$, we only have to consider a finite number of occurrences of $\langle\rangle$, which implies a finite number of constraints. We therefore only have to consider a finite number of subsets of the *event space*, built from the *solution space* of the combination of that finite number of constraints. The solution event can consequently be computed in finite time. Moreover, since all constraints in the construction of the solution event are finite-dimensional (*finite-dimensional-constraints condition*), the solution event is finite as well. $\square$

Proof of Lemma 4.3. We show that there is at least one element $P_{\mathbf{V}}$ in $\mathbf{P_V}$ by constructing it from the finite-dimensional $\mathbf{P}^k_{\mathbf{V}}$. The proof is by induction, assuming the number of *random variables* is infinite. Obviously the proof also holds for a finite number of random variables. We show that (1) there is at least one $P^1_{\mathbf{V}}$ in $\mathbf{P}^1_{\mathbf{V}}$ given an arbitrary $\mathbf{C}_1$ and (2) that given a $P^k_{\mathbf{V}}$ consistent with $\mathbf{C}_k$ there is always a *compatible* $P^{k+1}_{\mathbf{V}}$ consistent with $\mathbf{C}_{k+1}$ given that $\mathbf{C}_k$ and $\mathbf{C}_{k+1}$ are compatible. That $P^k_{\mathbf{V}}$ and $P^{k+1}_{\mathbf{V}}$ are compatible means that for any k-dimensional *event* e: $P^{k+1}_{\mathbf{V}}(\pi_{k+1}(e)) = P^k_{\mathbf{V}}(e)$. We therefore get an infinite sequence of compatible, finite-dimensional probability distributions from which we can construct a *probability measure* on the entire event space, given the basic assumptions we made in the lemma.

In the following, we do not give a complete construction of probability measures, because there are in general multiple distributions with the required probability. Therefore, we only fix the probability of a number of disjoint events in a way that makes it possible to construct a valid probability measure. We assign all probability mass to those disjoint events and no probability mass to the rest of the *probability space*. All possible measures assigning probabilities to events in this way clearly assign 1 to the entire space, are *countably additive* and therefore valid probability measures.

(1) The disjoint events we use to construct $P^1_{\mathbf{V}}$ are the intersections and relative complements of the events of $\mathbf{C}_1$, except the empty set. We then make sure that all those events are disjoint by restricting them to the minimal ones, i.e. the ones who have no subset in the collection. Intuitively, this can be seen as all areas of a *Venn diagram* which do not include other areas. We denote those events by $f_1, \ldots, f_n$ and assign for each $(p_i, e_i) \in \mathbf{C}_1$ probability p_i to an $f_j \subseteq e_i$. There is always at least one such f_j. In case we choose the same f_j for multiple e_i we sum the probabilities.

What remains to be shown is that a measure $P^1_{\mathbf{V}}$ constructed in this way obeys the inequalities of Definition 4.7, i.e. that for all events e:

$$\sum_{\substack{d \subseteq e \\ (p,d) \in \mathbf{C}_1}} p \leq P^1_{\mathbf{V}}(e) \leq \sum_{\substack{d \cap e \neq \emptyset \\ (p,d) \in \mathbf{C}_1}} p$$

First we prove that the lower bound holds. For each event e we consider all subsets of e in $\mathbf{C}_1$: $e_1, \ldots, e_m$. In the construction of $P^1_{\mathbf{V}}$ all probability mass assigned to all such $e_1, \dots, e_m$ must be assigned to subsets $f_1, \dots, f_{m'}$ of those $e_1, \dots, e_m$, which are in turn subsets of e as well. Therefore:

$$\sum_{\substack{d \subseteq e \\ (p,d) \in \mathbf{C}_1}} p \leq P^1_{\mathbf{V}}(e)$$

Similarly, the probability mass of each e_i for which $e_i \cap e = \emptyset$ can only be assigned to an f_j for which $f_j \cap e = \emptyset$, which means:

$$\sum_{\substack{d \cap e = \emptyset \\ (p,d) \in \mathbf{C}_1}} p \leq 1 - P^1_{\mathbf{V}}(e) \iff P^1_{\mathbf{V}}(e) \leq \sum_{\substack{d \cap e \neq \emptyset \\ (p,d) \in \mathbf{C}_1}} p$$

(2) We assume that we have given a $P^k_{\mathbf{V}}$ obeying the specification $\mathbf{C}_k$ and a $\mathbf{C}_{k+1}$ compatible with $\mathbf{C}_k$. From this we construct a probability measure $P^{k+1}_{\mathbf{V}}$, which is compatible with $P^k_{\mathbf{V}}$.

For each (p, e) in $\mathbf{C}_k$ there are $(p_1, e_1), \dots, (p_n, e_n)$ in $\mathbf{C}_{k+1}$, for which each $\pi_k(e_i) = e$, $1 \leq j \leq n$ and the sum of all p_i equals p, because $\mathbf{C}_k$ and $\mathbf{C}_{k+1}$ are compatible. Further, for each e there is an f to which the probability mass p was assigned in the construction of $P^k_{\mathbf{V}}$. We assign probability masses $p_1, \dots, p_n$ to the intersections and relative complements of $e_1 \cap f, \dots, e_n \cap f$. Such intersections are disjoint and non-empty and we denote them with $f_1, \dots, f_m$.

We have to make sure $P^{k+1}_{\mathbf{V}}$ is compatible with $P^k_{\mathbf{V}}$. For each k-dimensional f there are a number of $k+1$-dimensional $f_1, \dots, f_m$ and for all those events $P^k_{\mathbf{V}}(\pi_k(f_j)) = P^k_{\mathbf{V}}(f)$. Therefore, it is possible to construct $P^{k+1}_{\mathbf{V}}$ in such a way that it assigns the same probabilities to all g^{k+1} where $g \subseteq f$ as $P^k_{\mathbf{V}}$ assigns to g. This makes both measures compatible, since all f_i cover the entire probability space to which any probability mass is assigned.

The constructed probability measure obeys the bounds defined in Definition 4.7 with similar arguments as for $P^1_{\mathbf{V}}$. For each event e_i in $\mathbf{C}_{k+1}$ there are disjoint events $f_1, \dots, f_m$ to which some probability mass is assigned. All events in $\mathbf{C}_{k+1}$ which are subsets of any $f_1, \dots, f_m$ certainly contribute to the probability of e, while disjoint events do certainly not contribute to it. □

Proof of Theorem 4.1. The theorem follows directly from Lemma 4.3, Definition 4.3 and Definition 4.4. □

Proof of Proposition 4.2. Bounds for the probabilities of any event are defined by Definition 4.7. The theorem above puts those bounds on the query's solution event $\mathrm{SE}(q)$, whose probability is equal to the query's probability (Lemma 4.1). To show that the bounds are actually the minimum and maximum of the set of all probabilities, and therefore the most tight bounds, we show for each query q that there is a probability distribution in $\mathbf{P}$ such that the probability of q equals the lower (1) and a distribution such that the probability equals the upper bound (2).

We proof this by a construction similar to the proof of Lemma 4.3. The construction below applies to the construction of $P^1_{\mathbf{V}}$ as well as to the constructions

of $P_{\mathbf{V}}^{k+1}$. We assign probability masses to disjoint events $f_1, \ldots, f_n$ for each e in $\mathbf{C}_i$, but we do not assign the probability mass to arbitrary events as in the proof of Lemma 4.3. The events $f_1, \ldots, f_n$ are as before the minimal intersections and relative complements of all events in the *credal set specification*. Additionally, we split events f_i which are not subset of or disjoint with the solution event into two: $f_i \cap \mathrm{SE}(q)$ and $f_i \setminus \mathrm{SE}(q)$.
(1) For each e in $\mathbf{C}_i$, which is not a subset of the solution event, we assign the probability mass to an event from $f_1, \ldots, f_n$, which is disjoint with the solution event. Such event always exists, since all $f_1, \ldots, f_n$ which are subset of e, but not of $\mathrm{SE}(q)$, are split into a part which is disjoint with the solution event and a part which is not. This means only es which are subsets of the solution event contribute to the probability, which consequently equals the lower bound as defined by the proposition.
(2) We do the same kind of construction, but this time assign always to events which are subsets of the solution event if possible. This leaves out all events in $\mathbf{C}_i$, which are disjoint with the solution event and the probability equals the upper bound as defined by the proposition. □

Proof of Theorem 4.2. Using Proposition 4.2, we get the bounds given a finite-dimensional $\mathbf{C}_k$. If we increase k, some of the events e in $\mathbf{C}_k$ may split into subsets of e. Events e of which the probability contributes to the lower bound in k dimensions will do so as well in higher dimensions, since all events, e is possibly split into, are subsets of e. Some subsets of events, which do not contribute to the lower bound in k dimensions, may do so in higher dimensions. This means the lower bound is never overestimated for finite k and can only come closer to the actual lower bound for higher dimensions. Similarly, the upper bound is never underestimated for finite k and with increasing k the probability comes closer to the actual upper bound.

In case the bounds are underestimated or overestimated, this is always caused by the fact some event will be split for higher k, so for each event this can be compensated by increasing to that higher k. This means one can get arbitrarily close to the actual bounds. □

Proof of Corollary 4.1. This follows directly from Theorem 4.2, the sum rule of limits, the fact that probabilities of credal set specifications sum up to 1.0 (Definition 4.6) and the fact that $\mathrm{SE}(q)$ is disjoint with $\mathrm{SE}(\neg q)$ and $\mathrm{SE}(q) \cup \mathrm{SE}(\neg q) = \Omega_{\mathbf{V}}$. □

Proof of Proposition 4.3. We prove the equation for the upper bound. The proof for the lower bound is similar. We know that $P(q \mid e) = P(q \wedge e)/P(e)$, therefore $\overline{P}(q \mid e) = \overline{P}(q \wedge e)/Z$ where Z is the partition function which maximises $\overline{P}(q \mid e)$. The same partition function minimises $\underline{P}(\neg q \mid e)$, therefore $\underline{P}(\neg q \mid e) = \underline{P}(\neg q \wedge e)/Z$. By equivalence transformations and substitution of Z we get $\overline{P}(q \wedge e)\underline{P}(\neg q \mid e) = \underline{P}(\neg q \wedge e)\overline{P}(q \mid e)$. By using Corollary 4.1 ($\underline{P}(\neg q \mid e) = 1 - \overline{P}(q \mid e)$) we can substitute $\underline{P}(\neg q \mid e)$ and get $\overline{P}(q \wedge e)(1 - \overline{P}(q \mid e)) = \underline{P}(\neg q \wedge e)\overline{P}(q \mid e)$, which is equivalent to the equation in the lemma. □

Proof of Theorem 4.3. In the proof of Proposition 4.1 we show that we can determine a finite-dimensional solution event in case the finite-relevant-constraints condition and the finite-dimensional-constraints condition hold.

The bounds can be computed using Proposition 4.2. The equations in Proposition 4.2 require summing over all elements of a $\mathbf{C}_k$. This k has to be chosen such that all variables restricted by the solution event are included. A finite k can be found, because the finite-dimensional-constraints condition is fulfilled.

We finally can decide in finite time whether the event contained in each element of the finite-dimensional credal set specification is disjoint with or a subset of the solution event (*disjoint-events-decidability condition*). □

Proof of Lemma 4.4. This follows directly from the fact that the finite credal set specifications are compatible and Lemma 4.3. □

Proof of Proposition 4.4. The proof is a variation of the proof of Proposition 4.2. Disjointness of events corresponds to unsatisfiability of the corresponding constraints' conjunction and that an event is a subset of another corresponds to satisfiability of the implication $(\varphi_1 \rightarrow \varphi_2)$, which is equivalent to unsatisfiability of $\varphi_1 \land \neg\varphi_2$. □

Proof of Corollary 4.2. This follows from Definition 4.12 and Proposition 4.4. □

Proof of Corollary 4.3. This follows directly from Theorem 4.3, because the disjoint-events-decidability condition corresponds to decidable satisfiability of constraints, as shown in Proposition 4.4. □

Proof of Theorem 5.1. We focus on the problem of deciding whether $\underline{P}(\varphi) > p$, with probability $0 \leq p \leq 1$. To prove membership in PP^{C}, we show that this can be decided by a *probabilistic Turing machine* $\mathcal{M}$ in polynomial time, given an oracle for checking satisfiability of constraints. $\mathcal{M}$ computes the joint probability over choices and constraints in φ. First, $\mathcal{M}$ iterates over each variable $\mathbf{V}_i$ and chooses a particular constraint for that variable conform the probability mass associated to that constraint. Each computation path then corresponds to a specific choice Ψ for **Prog**. Then, $\mathcal{M}$ computes $\mathit{check}(\Psi \land \neg\varphi) = \mathit{unsat}$. If true, the state is accepted with probability $\frac{1}{2} + (1-p) \cdot \epsilon$, and with probability $\frac{1}{2} - p \cdot \epsilon$ otherwise. The probability of entering an accepting state is therefore $\underline{P}(\varphi) \cdot (\frac{1}{2} + (1-p)\epsilon) + (1 - \underline{P}(\varphi)) \cdot (\frac{1}{2} - p \cdot \epsilon) = \frac{1}{2} + \underline{P}(\varphi) \cdot \epsilon - p \cdot \epsilon$. Now, the probability of arriving in an accepting state is strictly larger than $\frac{1}{2}$ if and only if $\underline{P}(\varphi) > p$. □

Proof of Theorem 5.2. We assume that we have given a *tree decomposition* with *treewidth* t and use it to determine a variable order. Suppose we start at an arbitrary node s of the tree and make *case distinctions* for all variables in that node. The number of resulting *conjunctive normal forms* (CNFs) is bounded by d^t, as the size of the node is bounded by t and for each random variable at most d choices can be made. Furthermore, the number of choice constraints, each of those CNFs is conditioned on, is bounded by t. The complexity of the computation is therefore $O(t\, d^t)$, if we do not consider the recursive calls to OGWMC for the decomposed sub-CNFs. The reason why we do not consider recursive calls in first instance, is that not all sub-CNFs are different and we can exploit that using caching (Line 3 of Algorithm 2). We make use of the fact that the number of distinct sub-CNFs is bounded.

All sub-CNFs passed to further recursive calls do not include any of the variables of s. Either they are eliminated by simplification or they were included in an *imprecise constraint*. In the latter case, for each disjunction in which such imprecise constraints are included, choices for all random variables included are made and the optimisation in Lines 6–8 of Algorithm 2 can be applied. For each sub-CNF passed to recursive calls, a new tree decomposition can be assigned, after *decomposition* is applied (Line 13 of the algorithm). We eliminate s and all variables in s from the tree and get a number of disconnected subtrees. To each sub-CNF exactly one such subtree can be assigned, such that the same random variables are included in the sub-CNF as well as in the subtree. The subtrees do not share variables (Property 3 of Definition 5.3) and each random variable still remaining in the passed sub-CNFs is present in one of such subtree (Property 1 of Definition 5.3). This means that each such random variable is present in **exactly** one subtree. Furthermore, for each sub-CNF there is exactly one tree decomposition including all variables in that sub-CNF, because of Property 2 of Definition 5.3 and the definition of the *constraint primal graph* (Definition 5.2). We therefore have a single tree decomposition available to determine the variable order for each recursive call.

Because we apply caching, the number of consecutive computations is bounded by the number of different inputs of the algorithm. We only have to consider the number of distinct CNFs, since all variables occurring in choices have been removed, as discussed before. For each of such subtrees the number of distinct corresponding CNFs is bounded by d^m, where m is the number of variables the subtree and the node n chosen in the previous step have in common. If we now for each such CNF make case distinctions for all variables in the node of the subtree, originally connected to n, we have at most $t - m$ variables remaining. Therefore, the complexity for each single CNF is $O(t\, d^{t-m})$ and we have at most d^m different CNFs, which means that the complexity for all CNFs associated to that subtree is $O(t\, d^t)$.

This can be repeated, until all m tree nodes are eliminated. At least then computation terminates (Lines 4, 5, or 16 of the algorithm). We can conclude that the total complexity is $O(m\, t\, d^t)$. □

Proof of Lemma 6.1. We first show that a *partially evaluated hybrid probability tree* (PHPT) represents a partition of the entire *sample space* Ω, which is the product of all random variables' supports. To each node n of the PHPT we associate a subset of Ω:

$$\Omega_n = \bigcup_{\mathbf{X} \in \mathcal{V}} \text{range}(n, \mathbf{X}),$$

where $\mathcal{V}$ is the set of all random variables. Obviously, the sample space of the root node equals the problem's sample space Ω. At each branch of the tree, the space is split into two subspaces, which are exclusive and exhaustive (Definition 6.1). Therefore, the set of all leaves' sample spaces forms a partition of Ω.

The probability of space $P(\Omega_n)$, is the product of the probabilities of the path to n, because of the *independence* assumption we make.

The probability lower bound $\underline{P}(e)$, can also be written as sum of the probabilities of the space, associated to all leaf nodes labelled with $\top$. The PHPT just provides a more compact representation of this sum. Then we have:

$$\underline{P}(e) = \sum_{n \in T} P(\Omega_n),$$

where T is the set of all leaf nodes which are labelled with $\top$. We know that each Ω_n is a subset of e or in a *logical* sense that the formula representing e is a *consequence* of the statement that the space is restricted to Ω_n. From this and the fact that the spaces of all nodes in T are part of a partition of Ω, we can conclude:

$$\underline{P}(e) = \sum_{n \in T} P(\Omega_n) \leq P(e)$$

For the upper bound we consider the set of all leaf-nodes labelled with $\bot$ for which we have:

$$\overline{P}(e) = 1 - \sum_{n \in F} P(\Omega_n)$$

For each $n \in F$ we know that $\Omega_n \cap e = \emptyset$, therefore:

$$\sum_{n \in F} P(\Omega_n) + P(e) \leq 1 \iff P(e) \leq \overline{P}(e)$$

□

Proof of Lemma 6.2. We can first eliminate all boolean variables by making a PHPT that splits into all possible value assignments for such variables.

Each path that is not terminated by a $\top$ or $\bot$ contributes to the error. We can always reduce such error by evaluating the path further and can find a bound n of the number of levels we have to evaluate each branch further until we can terminate at least one path. The bound n depends on the problem: if we split each variable, certainly at least one primitive constraint can be reduced by $\top$ or $\bot$. As we have a finite number of random variables, as well as a finite number of primitive constraints in the formula, we can therefore find a finite bound n.

If p_ϵ is the probability mass of the original non-terminated path, we can find a $f \leq 1.0$ such that $f p_\epsilon \geq p_t$, where p_t is the probability mass of the terminated path found. Such f exists, because we can at each split, split into parts with equal probability mass, so $f = 0.5^n$. This means that the error, that the further evaluated sub-tree contributes, is at most $1 - f p_\epsilon$. As the infinite product of

$1 - f$ converges to 0 for a constant f and there are only a finite number of paths contributing to the error in each PHPT, also the total error converges to 0, if we evaluate all of those finite branches further.

The proof can straightforwardly be extended for infinite and discrete as well as for mixtures of discrete and continuous distributions. □

Proof of Proposition 6.1. This is a consequence of Lemma 6.2 and Proposition 4.3. □

Proof of Theorem 6.1. Given the proof of Lemma 6.2, the first requirement for termination is that we eventually continue evaluating all paths, of both PHPTs. The heuristics to choose the PHPT (*choose_phpt*) and node (*choose_node*) with the maximal error ensure that, as the error converges to 0. So the error of each branch will become eventually smaller than that of any other, when evaluated further. The variable choice heuristic (*choose_rvar*) ensures that within a bounded number of steps, always a terminating node is found. The same variable cannot be chosen again, before a primitive constraint in which the variable occurs cannot be eliminated and only variables occurring in still non-eliminated constraints can be chosen. Finally, the partition heuristics (*choose_part*) ensures a lower bound on how much the error is reduced, as the probability mass is either split, using a heuristic which is more effective than splitting into two equal parts (as in the proof of Lemma 6.2), or otherwise a primitive constraint is eliminated, which reduces the error even more. □

Proof of Lemma 7.1. As all heads occur only positively in all bodies, decreasing the probability of a rule being true can only decrease the probability of all possible queries. Therefore choosing the lower probability bounds for all rules, results in all query probabilities becoming the lower bounds of the probability set in Definition 7.2. □

Proof of Lemma 7.2. The lemma obviously holds as replacing imprecisions with point probabilities is a linear transformation, not changing the complexity of *inference*. □

Proof of Proposition 7.1. We have already shown that the probabilities for heads with rule-imprecision is correct (Lemma 7.1). What remains to be shown is that the translation of head-imprecisions to rule-imprecisions in Algorithm 4 is correct as well. For that we have to show that the lower probability of the translated rule, in case a set of defining *atoms* $\mathbf{B}$ are true and all others not, equals $lower(\mathbf{P}_h(\mathbf{B}))$, as defined in Equation 7.3. The probability of any rule with body $\mathbf{B}' \subseteq \mathbf{B}$ is given by $subset\_lower_h(\mathbf{B}')$. In total, the probability of h given $\mathbf{B}$ is the *noisy-or* combination of all those probabilities, which is:

$$1 - \Big(\prod_{\mathbf{B}' \subseteq \mathbf{B}} 1 - subset\_lower_h(\mathbf{B}') \Big)$$

We denote the set of all atoms in the bodies of rules defining head h with $\mathbf{B}_h$. We then have:

$$\begin{aligned}
&P_{\mathbf{R}}(h \mid \wedge_{a \in \mathbf{B}_h} \text{ if } a \in \mathbf{B} \text{ then } a \text{ else } \neg a)\\
&\quad = 1 - \Big(\prod_{\mathbf{B}' \subseteq \mathbf{B}} 1 - \mathit{subset\_lower}_h(\mathbf{B}')\Big)\\
&\quad = 1 - \Big(\prod_{\mathbf{B}' \subset \mathbf{B}} 1 - \mathit{subset\_lower}_h(\mathbf{B}')\Big)\Big(1 - \mathit{subset\_lower}_h(\mathbf{B})\Big)\\
&\quad = 1 - \left(\prod_{\mathbf{B}' \subset \mathbf{B}} 1 - \mathit{subset\_lower}_h(\mathbf{B}')\right)\left(1 - \left(1 - \frac{1 - \mathit{lower}\big(\mathbf{P}_h(\mathbf{B})\big)}{\prod_{\mathbf{B}' \subset \mathbf{B}} 1 - \mathit{subset\_lower}_h(\mathbf{B}')}\right)\right)\\
&\quad = 1 - \left(\prod_{\mathbf{B}' \subset \mathbf{B}} 1 - \mathit{subset\_lower}_h(\mathbf{B}')\right)\left(\frac{1 - \mathit{lower}\big(\mathbf{P}_h(\mathbf{B})\big)}{\prod_{\mathbf{B}' \subset \mathbf{B}} 1 - \mathit{subset\_lower}_h(\mathbf{B}')}\right)\\
&\quad = 1 - 1 + \mathit{lower}\big(\mathbf{P}_h(\mathbf{B})\big)\\
&\quad = \mathit{lower}\big(\mathbf{P}_h(\mathbf{B})\big)
\end{aligned}$$

□

Proof of Theorem 7.1. This follows from the fact that the correct (Proposition 7.1) translation of Algorithm 4 has polynomial complexity and does not change the program's treewidth. □

BIBLIOGRAPHY

[1] Martin Abadi and Joseph Y Halpern. Decidability and expressiveness for first-order logics of probability. *Information and computation*, 112(1):1–36, 1994.

[2] Nicos Angelopoulos. clp(pdf(y)): Constraints for probabilistic reasoning in logic programming. In Francesca Rossi, editor, *Principles and Practice of Constraint Programming 2003*, volume 2833 of *Lecture Notes in Computer Science*, pages 784–788. Springer Berlin Heidelberg, 2003. ISBN 978-3-540-20202-8.

[3] Krzysztof R Apt, Howard A Blair, and Adrian Walker. *Towards a theory of declarative knowledge*. IBM TJ Watson Research Center, 1986.

[4] Stefan Arnborg, Derek G Corneil, and Andrzej Proskurowski. Complexity of finding embeddings in ak-tree. *SIAM Journal on Algebraic Discrete Methods*, 8(2):277–284, 1987.

[5] Nimar S. Arora, Rodrigo de Salvo Braz, Erik B. Sudderth, and Stuart J. Russell. Gibbs sampling in open-universe stochastic languages. In *UAI 2010, Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence, Catalina Island, CA, USA, July 8-11, 2010*, pages 30–39, 2010.

[6] F. Bacchus. *Representing and Reasoning with Probabilistic Knowlege: A Logical Approach to Probabilities*. MIT Press, Cambridge, MA, 1990.

[7] Roberto J. Bayardo and Joseph Daniel Pehoushek. Counting models using connected components. In *AAAI/IAAI*, pages 157–162, 2000.

[8] Vaishak Belle, Andrea Passerini, and Guy Van den Broeck. Probabilistic inference in hybrid domains by weighted model integration. In *Proceedings of 24th International Joint Conference on Artificial Intelligence (IJCAI)*, 2015.

[9] Vaishak Belle, Guy Van den Broeck, and Andrea Passerini. Hashing-based approximate probabilistic inference in hybrid domains. In *Proceedings of the 31st Conference on Uncertainty in Artificial Intelligence (UAI)*, 2015.

[10] Jakob Bernoulli. *Ars conjectandi*. Impensis Thurnisiorum, fratrum, 1713.

[11] John Binder, Daphne Koller, Stuart Russell, Keiji Kanazawa, and Padhraic Smyth. Adaptive probabilistic networks with hidden variables. In *Machine Learning*, pages 213–244, 1997.

[12] Hans L Bodlaender. A linear time algorithm for finding tree-decompositions of small treewidth. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 226–234. ACM, 1993.

[13] Hans L. Bodlaender. A tourist guide through treewidth. *Technical report RUU-CS*, 92, 1993.

[14] George Boole. *The mathematical analysis of logic*. Philosophical Library, 1847.

[15] George Boole. The laws of thought (1854). 1854.

[16] Craig Boutilier, Nir Friedman, Moises Goldszmidt, and Daphne Koller. Context-specific independence in Bayesian networks. In *Proceedings of the Twelfth international conference on Uncertainty in artificial intelligence*, pages 115–123. Morgan Kaufmann Publishers Inc., 1996.

[17] Steve Brooks, Andrew Gelman, Galin Jones, and Xiao-Li Meng. *Handbook of Markov Chain Monte Carlo*. CRC Press, 2011.

[18] Randal E Bryant. Graph-based algorithms for boolean function manipulation. *Computers, IEEE Transactions on*, 100(8):677–691, 1986.

[19] Bruce G Buchanan, Edward Hance Shortliffe, et al. *Rule-based expert systems*, volume 3. Addison-Wesley Reading, MA, 1984.

[20] Andrés Cano, José E Cano, and Serafín Moral. Convex sets of probabilities propagation by simulated annealing. In *In Proceedings of the Fith International Conference IPMU'94*. Citeseer, 1994.

[21] R.N. Carvalho, P.C.G. Costa, K.B. Laskey, and K.C. Chang. PROGNOS: Predictive situational awareness with probabilistic ontologies. In *Proc. of the 13th Conference on Information Fusion*, pages 1–8, 2010.

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

[23] C.-H. Chen, L.P. Khoo, Y.T. Chong, and X.F. Yin. Knowledge discovery using genetic algorithm for maritime situational awareness. *Expert Systems with Applications*, 41(6):2742–2753, 2014.

[24] Dmitry Chistikov, Rayna Dimitrova, and Rupak Majumdar. Approximate counting in smt and value estimation for probabilistic programs. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 320–334. Springer, 2015.

[25] Jaesik Choi, Eyal Amir, and David J Hill. Lifted inference for relational continuous models. In *UAI*, volume 10, pages 126–134, 2010.

[26] K.L. Clark. Negation as failure. *Logic and Data Bases*, pages 293–322, 1978.

[27] P. Codognet and D. Diaz. Compiling Constraints in CLP(FD). *J. Log. Program.*, 27(3):185–226, 1996.

[28] Paulo C. G. Costa, Kathryn B. Laskey, Kuo-Chuand Chang, Weiand Sun, Cheol Y. Park, and Shou Matsumoto. High-level information fusion with Bayesian Semantics. In *Proc. of the 9th Bayesian Modelling Applications Workshop, held at the UAI*, 2012.

[29] Vıtor Santos Costa and Aline Paes. On the relationship between PRISM and CLP($\mathcal{BN}$). In *Proc. of the Int. Workshop on Statistical Relational Learning (SRL-2009)*, 2009.

[30] Vítor Santos Costa, David Page, Maleeha Qazi, and James Cussens. CLP (BN): Constraint logic programming for probabilistic knowledge. In *Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence*, pages 517–524. Morgan Kaufmann Publishers Inc., 2002.

[31] Fabio G Cozman. Credal networks. *Artificial Intelligence*, 120(2):199–233, 2000.

[32] Fabio Gagliardi Cozman, Cassio Polpo De Campos, Jaime Shinsuke Ide, and José Carlos Ferreira da Rocha. Propositional and relational Bayesian networks associated with imprecise and qualitative probabilistic assessments. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 104–111. AUAI Press, 2004.

[33] Adnan Darwiche. On the tractable counting of theory models and its application to truth maintenance and belief revision. *Journal of Applied Non-Classical Logics*, 11(1-2):11–34, 2001.

[34] Adnan Darwiche. Recursive conditioning. *Artificial Intelligence*, 126(1):5–41, 2001.

[35] Adnan Darwiche. New advances in compiling CNF to decomposable negation normal form. In *Proc. of ECAI*, pages 328–332. Citeseer, 2004.

[36] James H Davenport and Joos Heintz. Real quantifier elimination is doubly exponential. *Journal of Symbolic Computation*, 5(1):29–35, 1988.

[37] Cassio Polpo De Campos and Fabio Gagliardi Cozman. The inferential complexity of Bayesian and credal networks. In *IJCAI*, volume 5, pages 1313–1318, 2005.

[38] Abraham De Moivre. *The doctrine of chances: or, A method of calculating the probabilities of events in play*, volume 1. Chelsea Publishing Company, 1756.

[39] Leonardo De Moura and Nikolaj Bjørner. Z3: An efficient SMT solver. In *Proceedings of the Theory and Practice of Software, 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, TACAS'08/ETAPS'08, pages 337–340, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 3-540-78799-2, 978-3-540-78799-0. URL `http://dl.acm.org/citation.cfm?id=1792734.1792766`.

[40] Luc De Raedt and Angelika Kimmig. Probabilistic (logic) programming concepts. *Machine Learning*, pages 1–43, 2015.

[41] Rodrigo de Salvo Braz, Ciaran O'Reilly, Vibhav Gogate, and Rina Dechter. Probabilistic Inference Modulo Theories. In *Workshop on Hybrid Reasoning at IJCAI 2015*, 2015.

[42] Rina Dechter. Reasoning with probabilistic and deterministic graphical models: Exact algorithms. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 7(3):1–191, 2013.

[43] Reinhard Diestel. *Graph theory*, volume 173 of *Graduate Texts in Mathematics*. Springer-Verlag, Berlin, third edition, 2005. ISBN 978-3-540-26182-7; 3-540-26182-6; 978-3-540-26183-4.

[44] Pedro Domingos, Stanley Kok, Daniel Lowd, Hoifung Poon, Matthew Richardson, and Parag Singla. Markov logic. In *Probabilistic Inductive Logic Programming*, pages 92–117, 2008.

[45] Peter J Downey, Ravi Sethi, and Robert Endre Tarjan. Variations on the common subexpression problem. *Journal of the ACM (JACM)*, 27(4):758–771, 1980.

[46] Bruno Dutertre and Leonardo De Moura. The yices SMT solver. Technical report, Computer Science Laboratory, SRI International, 2006.

[47] Enrico Fagiuoli and Marco Zaffalon. 2u: an exact interval propagation algorithm for polytrees with binary variables. *Artificial Intelligence*, 106(1): 77–107, 1998.

[48] Daan Fierens, Guy Van den Broeck, Joris Renkens, Dimitar Shterionov, Bernd Gutmann, Ingo Thon, Gerda Janssens, and Luc De Raedt. Inference and learning in probabilistic logic programs using weighted Boolean formulas. *Theory and Practice of Logic Programming*, 15(03):358–401, 2015.

[49] Yvonne Fischer and Jürgen Beyerer. A top-down-view on intelligent surveillance systems. In *Proc. of the 7th International Conference on Systems*, pages 43–48, Saint Gilles, Reunion, February 2012.

[50] Gottlob Frege and Claude Imbert. *Écrits logiques et philosophiques*. Ed. du Seuil, 1971.

[51] Haim Gaifman et al. Concerning measures on boolean algebras. *Pacific J. Math*, 14:61–73, 1964.

[52] Jesús García, José Luis Guerrero Madrid, Álvaro Luis Bustamante, and José M Molina. Robust sensor fusion in real maritime surveillance scenarios. In *Proc. of the 13th Conference on Information Fusion*, 2010.

[53] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In *ICLP/SLP*, volume 88, pages 1070–1080, 1988.

[54] L. Getoor and B. Taskar, editors. *An Introduction to Statistical Relational Learning*. MIT Press, 2007.

[55] Kurt Gödel. Die vollständigkeit der axiome des logischen funktionenkalküls. *Monatshefte für Mathematik*, 37(1):349–360, 1930.

[56] Robert Goldblatt. *Lectures on the hyperreals: an introduction to nonstandard analysis*, volume 188. Springer, 1998.

[57] ND Goodman, VK Mansinghka, D Roy, K Bonawitz, and JB Tenenbaum. Church: A language for generative models. In *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence, UAI 2008*, pages 220–229, 2008.

[58] Erich Grädel. On transitive closure logic. In *Computer Science Logic: 5th Workshop, CSL'91*, volume 626, pages 149–163. Springer, 1992.

[59] M. Guerriero, P. Willett, S. Coraluppi, and C. Carthel. Radar/AIS data fusion and SAR tasking for maritime surveillance. In *Proc. of the 11th Conference on Information Fusion*, pages 1–5, 2008.

[60] Bernd Gutmann, Manfred Jaeger, and Luc De Raedt. Extending ProbLog with continuous distributions. In Paolo Frasconi and Francesca Alessandra Lisi, editors, *Proc. of the 20th International Conference on Inductive Logic Programming (ILP–10)*, Firenze, Italy, 2010.

[61] Bernd Gutmann, Ingo Thon, Angelika Kimmig, Maurice Bruynooghe, and Luc De Raedt. The magic of logical inference in probabilistic programming. *Theory and Practice of Logic Programming*, 11:663–680, 2011.

[62] Theodore Hailperin. *Boole's logic and probability*. North Holland Amsterdam, 1976.

[63] Joseph Y Halpern. An analysis of first-order logics of probability. *Artificial intelligence*, 46(3):311–350, 1990.

[64] Aviad Heifetz and Philippe Mongin. The modal logic of probability. In *Proceedings of the 7th conference on Theoretical aspects of rationality and knowledge*, pages 175–185. Morgan Kaufmann Publishers Inc., 1998.

[65] Teun Hendriks and Piërre van de Laar. Metis: Dependable cooperative systems for public safety. *Procedia Computer Science*, 16:542–551, 2013.

[66] P. Van Hentenryck. *Constraint Satisfaction in Logic Programming*. The MIT Press, 1989.

[67] Jacques Herbrand. Recherches sur la théorie de la démonstration. 1930.

[68] Mauricio A Hernández and Salvatore J Stolfo. The merge/purge problem for large databases. In *ACM SIGMOD Record*, volume 24, pages 127–138. ACM, 1995.

[69] Matthew D. Hoffman and Andrew Gelman. The no-u-turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15(1):1593–1623, 2014. URL http://dl.acm.org/citation.cfm?id=2638586.

[70] Marcus Hutter, John W Lloyd, Kee Siong Ng, and William TB Uther. Probabilities on sentences in an expressive logic. *Journal of Applied Logic*, 11(4): 386–420, 2013.

[71] Muhammad Asiful Islam, C. R. Ramakrishnan, and I. V. Ramakrishnan. Inference in probabilistic logic programs with continuous random variables. *Theory and Practice of Logic Programming*, 12(4-5):505–523, 2012.

[72] J. Jaffar and J.L. Lassez. Constraint logic programming. In *Proceedings of the 14th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*, pages 111–119. ACM, 1987.

[73] Joxan Jaffar, Spiro Michaylov, Peter J. Stuckey, and Roland H. C. Yap. The CLP(R) language and system. *ACM Trans. Program. Lang. Syst.*, 14(3):339–395, May 1992. ISSN 0164-0925. doi: 10.1145/129393.129398. URL http://doi.acm.org/10.1145/129393.129398.

[74] Joxan Jaffar, Michael J. Maher, Kim Marriott, and Peter J. Stuckey. The semantics of constraint logic programs. *The Journal of Logic Programming*, 37(1-3):1–46, 1998. doi: 10.1016/S0743-1066(98)10002-X. URL http://dx.doi.org/10.1016/S0743-1066(98)10002-X.

[75] Dominik Jain, Bernhard Kirchlechner, and Michael Beetz. Extending Markov Logic to Model Probability Distributions in Relational Domains. In *Lecture Notes in Computer Science*, volume 4667, pages 129–143. Springer, 2007. ISBN 978-3-540-74564-8.

[76] Tomi Janhunen. Representing normal programs with clauses. In *ECAI*, volume 16, page 358, 2004.

[77] Frank Jensen and S. Anderson. Approximations in Bayesian belief universe for knowledge based systems. In *Proceedings of the Sixth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-90)*, pages 162–169, Corvallis, Oregon, 1990. AUAI Press.

[78] J.F. De Paz J.M. Corchado, A. Mata and D. Del Pozo. A new cbr approach to the oil spill problem. In *ECAI*, pages 643–647, 2008.

[79] Narendra Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pages 302–311. ACM, 1984.

[80] Kristian Kersting. Lifted probabilistic inference. In *ECAI*, pages 33–38, 2012.

[81] Andreĭ Nikolaevich Kolmogorov. Foundations of the theory of probability. 1950.

[82] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[83] Helge Langseth, Thomas D Nielsen, Antonio Salmerón, et al. Mixtures of truncated basis functions. *International Journal of Approximate Reasoning*, 53 (2):212–227, 2012.

[84] K.B. Laskey. MEBN: A language for first-order Bayesian knowledge bases. *Artif. Intell.*, 172(2-3):140–178, 2008.

[85] Steffen L Lauritzen. Propagation of probabilities, means, and variances in mixed graphical association models. *Journal of the American Statistical Association*, 87(420):1098–1108, 1992.

[86] Hector J Levesque and Ronald J Brachman. *A fundamental tradeoff in knowledge representation and reasoning*. Laboratory for Artificial Intelligence Research, Fairchild, Schlumberger, 1984.

[87] Isaac Levi. The enterprise of knowledge. 1980.

[88] Lei Li, Bharath Ramsundar, and Stuart Russell. Dynamic scaled sampling for deterministic constraints. In *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*, pages 397–405, 2013.

[89] J.W. Lloyd. *Foundations of Logic Programming, 2nd Edition*. Springer, 1987.

[90] David Madigan, Krzysztof Mosurski, and Russell G Almond. Graphical explanation in belief networks. *Journal of Computational and Graphical Statistics*, 6(2):160–181, 1997.

[91] T. Mantadelis and G. Janssens. Dedicated tabling for a probabilistic setting. In *ICLP (Technical Communications)*, pages 124–133, 2010.

[92] Victor W Marek and Miroslaw Truszczyński. Stable models and an alternative logic programming paradigm. In *The Logic Programming Paradigm*, pages 375–398. Springer, 1999.

[93] Y. V. Matiyasevich. Diophantine representation of recursively enumerable predicates. In J.E. Fenstad, editor, *Second Scandinavian Logic Symposium*, pages 171–177. North-Holland Publishing Company, 1971.

[94] Andrew McCallum, Karl Schultz, and Sameer Singh. Factorie: Probabilistic programming via imperatively defined factor graphs. In *Advances in Neural Information Processing Systems*, pages 1249–1257, 2009.

[95] Steffen Michels, Marina Velikova, Arjen Hommersom, and Peter J.F. Lucas. A Probabilistic Logic–based Model for Fusing Attribute Information of Objects Under Surveillance. Technical Report ICIS–R12006, Radboud University Nijmegen, December 2012. `https://pms.cs.ru.nl/iris-diglib/src/getContent.php?id=2012-Michels-Fusion`.

[96] Steffen Michels, Arjen Hommersom, Peter J. F. Lucas, Marina Velikova, and Pieter W. M. Koopman. Inference for a new probabilistic constraint logic. In Francesca Rossi, editor, *IJCAI*. IJCAI/AAAI, 2013. ISBN 978-1-57735-633-2.

[97] Steffen Michels, Marina Velikova, Arjen Hommersom, and Peter JF Lucas. A decision support model for uncertainty reasoning in safety and security tasks. In *Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on*, pages 663–668. IEEE, 2013.

[98] Steffen Michels, Arjen Hommersom, Peter J. F. Lucas, and Marina Velikova. Imprecise probabilistic horn clause logic. In *ECAI 2014 - 21st European Conference on Artificial Intelligence, 18-22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014)*, pages 621–626, 2014. doi: 10.3233/978-1-61499-419-0-621. URL `http://dx.doi.org/10.3233/978-1-61499-419-0-621`.

[99] Steffen Michels, Arjen Hommersom, Peter J. F. Lucas, and Marina Velikova. A new probabilistic constraint logic programming language based on a generalised distribution semantics. *Artif. Intell.*, 228:1–44, 2015. doi: 10.1016/j.artint.2015.06.008. URL `http://dx.doi.org/10.1016/j.artint.2015.06.008`.

[100] Steffen Michels, Arjen Hommersom, and Peter J. F. Lucas. Approximate probabilistic inference with bounded error for hybrid probabilistic logic programming. Submitted to IJCAI'16, 2016.

[101] Brian Milch and Daphne Koller. Probabilistic models for agents' beliefs and decisions. In *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, pages 389–396. Morgan Kaufmann Publishers Inc., 2000.

[102] Brian Milch, Bhaskara Marthi, Stuart J. Russell, David Sontag, Daniel L. Ong, and Andrey Kolobov. BLOG: probabilistic models with unknown objects. In *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30-August 5, 2005*, pages 1352–1359, 2005. URL `http://www.ijcai.org/papers/1546.pdf`.

[103] Brian Christopher Milch. *Probabilistic models with unknown objects*. PhD thesis, Citeseer, 2006.

[104] Alvaro E Monge, Charles Elkan, et al. The field matching problem: Algorithms and applications. In *KDD*, pages 267–270, 1996.

[105] Joris M. Mooij and Hilbert J. Kappen. Sufficient conditions for convergence of loopy belief propagation. In *UAI '05, Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence, Edinburgh, Scotland, July 26-29, 2005*, pages 396–403, 2005.

[106] Christian Muise, Sheila A Mcilraith, J Christopher Beck, and Eric Hsu. DSHARP: Fast d-DNNF compilation with sharpSAT. In *Canadian Conference on Artificial Intelligence*, Canadian Conference on Artificial Intelligence, 2012.

[107] Kevin P Murphy, Yair Weiss, and Michael I Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 467–475. Morgan Kaufmann Publishers Inc., 1999.

[108] Kevin Patrick Murphy. *Dynamic bayesian networks: representation, inference and learning*. PhD thesis, University of California, Berkeley, 2002.

[109] M. Naderpour, J. Lu, and G. Zhang. An intelligent situation awareness support system for safety-critical environments. *Decision Support Systems*, in press, 2014.

[110] Jacques Neveu. *Mathematical Foundations of the Calculus of Probability*. Holden-Day, San Francisco, 1965.

[111] Raymond Ng and Venkatramanan Siva Subrahmanian. Probabilistic logic programming. *Information and computation*, 101(2):150–201, 1992.

[112] Ilkka Niemelä. Logic programs with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence*, 25 (3-4):241–273, 1999.

[113] N.J. Nilsson. Probabilistic logic. *Artif. Intell.*, 28(1):71–87, 1986.

[114] Davide Nitti, Tinne De Laet, and Luc De Raedt. A particle filter for hybrid relational domains. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, November 3-7, 2013*, pages 2764–2771, 2013. doi: 10.1109/IROS.2013.6696747. URL `http://dx.doi.org/10.1109/IROS.2013.6696747`.

[115] Mehmet A Orgun and Wanli Ma. An overview of temporal and modal logic programming. In *Temporal logic*, pages 445–479. Springer, 1994.

[116] Christos H Papadimitriou. On the complexity of integer programming. *Journal of the ACM (JACM)*, 28(4):765–768, 1981.

[117] Jeff Paris. Common sense and maximum entropy. *Synthese*, 117(1):75–93, 1998.

[118] Mark A Paskin. *Maximum entropy probabilistic logic*. Computer Science Division, University of California, 2002.

[119] Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 1988.

[120] Avi Pfeffer. Figaro: An object-oriented probabilistic programming language. *Charles River Analytics Technical Report*, page 137, 2009.

[121] David Poole. Logic programming, abduction and probability. *New Generation Computing*, 11(3-4):377–400, 1993.

[122] David Poole. The independent choice logic and beyond. In Luc De Raedt, Paolo Frasconi, Kristian Kersting, and Stephen Muggleton, editors, *Probabilistic inductive logic programming*, pages 222–243. Springer-Verlag, Berlin, Heidelberg, 2008. ISBN 3-540-78651-1, 978-3-540-78651-1. URL `http://dl.acm.org/citation.cfm?id=1793956.1793966`.

[123] David Poole and Nevin Lianwen Zhang. Exploiting contextual independence in probabilistic inference. *J. Artif. Intell. Res.(JAIR)*, 18:263–313, 2003.

[124] James Gary Propp and David Bruce Wilson. Exact sampling with coupled Markov chains and applications to statistical mechanics. *Random structures and Algorithms*, 9(1-2):223–252, 1996.

[125] Luc De Raedt, Angelika Kimmig, and Hannu Toivonen. ProbLog: a probabilistic prolog and its application in link discovery. In *Proc. of the 20th International Joint Conference on Artificial Intelligence*, pages 2468–2473. AAAI Press, 2007.

[126] Raymond Reiter. A logic for default reasoning. *Artificial intelligence*, 13(1): 81–132, 1980.

[127] S. Reizler. *Probabilistic Constraint Logic Programming*. PhD thesis, University of Tubingen, Tubingen, Germany, 1998.

[128] Joris Renkens, Angelika Kimmig, Guy Van den Broeck, and Luc De Raedt. Explanation-based approximate weighted model counting for probabilistic logics. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, 2014.

[129] Joris Renkens, Angelika Kimmig, and Luc De Raedt. Lazy explanation-based approximation for probabilistic logic programming. *arXiv preprint arXiv:1507.02873*, 2015.

[130] Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine Learning*, 62(1-2):107–136, 2006.

[131] John Alan Robinson. The generalized resolution principle. *Machine intelligence*, 3:77–93, 1968.

[132] Marko Samer and Stefan Szeider. Algorithms for propositional model counting. *Journal of Discrete Algorithms*, 8(1):50–64, 2010.

[133] Tian Sang, Fahiem Bacchus, Paul Beame, Henry A Kautz, and Toniann Pitassi. Combining component caching and clause learning for effective model counting. *SAT*, 4:7th, 2004.

[134] Tian Sang, Paul Beame, and Henry Kautz. Heuristics for fast exact model counting. In *Theory and Applications of Satisfiability Testing*, pages 226–240. Springer, 2005.

[135] Tian Sang, Paul Beame, and Henry Kautz. Solving Bayesian networks by weighted model counting. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05)*, volume 1, pages 475–482, 2005.

[136] Scott Sanner and Ehsan Abbasnejad. Symbolic variable elimination for discrete and continuous graphical models. In *AAAI*, 2012.

[137] Taisuke Sato. A statistical learning method for logic programs with distribution semantics. In *ICLP*, pages 715–729, 1995.

[138] Taisuke Sato and Yoshitaka Kameya. PRISM: a language for symbolic-statistical modeling. In *Proc. of the 15th International Joint Conference on Artificial Intelligence*, pages 1330–1335, 1997.

[139] Taisuke Sato and Yoshitaka Kameya. New advances in logic-based probabilistic modeling by prism. In *Probabilistic inductive logic programming*, pages 118–155. Springer, 2008.

[140] Taisuke Sato, Masakazu Ishihata, and Katsumi Inoue. Constraint-based probabilistic modeling for statistical abduction. *Machine learning*, 83(2): 241–264, 2011.

[141] Roeland Scheepens, Steffen Michels, Huub van de Wetering, and Jarke J. van Wijk. Rationale visualization for safety and security. *Computer Graphics Forum*, 34(3):191–200, 2015. ISSN 1467-8659. doi: 10.1111/cgf.12631. URL `http://dx.doi.org/10.1111/cgf.12631`.

[142] Glenn Shafer. *The art of causal conjecture*. MIT press, 1996.

[143] Afsaneh Shirazi and Eyal Amir. Probabilistic modal logic. In *Proceedings of the national conference on artificial intelligence*, volume 22, page 489. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007.

[144] L. Snidaro, I. Visentini, and K. Bryan. Fusing uncertain knowledge and evidence for maritime situational awareness via Markov Logic Networks. *Information Fusion*, in press, 2013.

[145] Alfred Tarski. The concept of truth in the languages of the deductive sciences. *Prace Towarzystwa Naukowego Warszawskiego, Wydzial III Nauk Matematyczno-Fizycznych*, 34:13–172, 1933.

[146] Sekhar C Tatikonda and Michael I Jordan. Loopy belief propagation and Gibbs measures. In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, pages 493–500. Morgan Kaufmann Publishers Inc., 2002.

[147] Son D. Tran and Larry S. Davis. Event modeling and recognition using Markov logic networks. In *Proc. of the 10th European Conference on Computer Vision: Part II*, pages 610–623, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 978-3-540-88685-3. doi: 10.1007/978-3-540-88688-4_45. URL http://dx.doi.org/10.1007/978-3-540-88688-4_45.

[148] R. Valdez, P.W. Yoon, T. Liu, and M.J. Khoury. Family history and prevalence of diabetes in the us population the 6-year results from the national health and nutrition examination survey (1999–2004). *Diabetes Care*, 30(10): 2517–2522, 2007.

[149] L. van der Gaag. *On Probability Intervals and Their Updating*. Technical report/ Department of Computer Science, University of Utrecht. Department, Univ., 1990. URL http://books.google.nl/books?id=nH9JGwAACAAJ.

[150] Allen Van Gelder, Kenneth A Ross, and John S Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM (JACM)*, 38(3): 619–649, 1991.

[151] Marina Velikova, Peter Novák, Bas Huijbrechts, Jan Laarhuis, Jesper Hoeksma, and Steffen Michels. An integrated reconfigurable system for maritime situational awareness. In *ECAI 2014 - 21st European Conference on Artificial Intelligence, 18-22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014)*, pages 1197–1202, 2014. doi: 10.3233/978-1-61499-419-0-1197. URL http://dx.doi.org/10.3233/978-1-61499-419-0-1197.

[152] Joost Vennekens, Marc Denecker, and Maurice Bruynooghe. CP-logic: A Language of Causal Probabilistic Events and Its Relation to Logic Programming. *Theory and practice of logic programming*, 9:245–308, 2009.

[153] Jonas Vlasselaer, Guy Van den Broeck, Angelika Kimmig, Wannes Meert, and Luc De Raedt. Anytime inference in probabilistic logic programs with tp-compilation. 2015.

[154] Peter Walley. Towards a unified theory of imprecise probability. *International Journal of Approximate Reasoning*, 24(2):125–148, 2000.

[155] Edward L. Waltz and James Llinas. *Multisensor Data Fusion*. Artech House, Inc., Norwood, MA, USA, 1990. ISBN 0890062773.

[156] Jue Wang and Pedro Domingos. Hybrid Markov logic networks. In *Proceedings of the 23rd national conference on Artificial intelligence - Volume 2*,

AAAI'08, pages 1106–1111. AAAI Press, 2008. ISBN 978-1-57735-368-3. URL http://dl.acm.org/citation.cfm?id=1620163.1620244.

[157] Kurt Weichselberger and Thomas Augustin. On the symbiosis of two concepts of conditional interval probability. In *ISIPTA*, volume 3, pages 608–629, 2003.

[158] Jon Williamson. Philosophies of probability. *Handbook of the Philosophy of Mathematics*, 4:493–533, 2009.

[159] Jonathan S Yedidia, William T Freeman, and Yair Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *Information Theory, IEEE Transactions on*, 51(7):2282–2312, 2005.

[160] Dong Yu, Li Deng, and Alex Acero. Using continuous features in the maximum entropy model. *Pattern Recognition Letters*, 30(14):1295–1300, 2009.

LIST OF SIKS DISSERTATIONS

2009

2009-01 Rasa Jurgelenaite (RUN)
Symmetric Causal Independence Models

2009-02 Willem Robert van Hage (VU)
Evaluating Ontology-Alignment Techniques

2009-03 Hans Stol (UvT)
A Framework for Evidence-based Policy Making Using IT

2009-04 Josephine Nabukenya (RUN)
Improving the Quality of Organisational Policy Making using Collaboration Engineering

2009-05 Sietse Overbeek (RUN)
Bridging Supply and Demand for Knowledge Intensive Tasks - Based on Knowledge, Cognition, and Quality

2009-06 Muhammad Subianto (UU)
Understanding Classification

2009-07 Ronald Poppe (UT)
Discriminative Vision-Based Recovery and Recognition of Human Motion

2009-08 Volker Nannen (VU)
Evolutionary Agent-Based Policy Analysis in Dynamic Environments

2009-09 Benjamin Kanagwa (RUN)
Design, Discovery and Construction of Service-oriented Systems

2009-10 Jan Wielemaker (UVA)
Logic programming for knowledge-intensive interactive applications

2009-11 Alexander Boer (UVA)
Legal Theory, Sources of Law & the Semantic Web

2009-12 Peter Massuthe (TUE, Humboldt-Universitaet zu Berlin)
Operating Guidelines for Services

2009-13 Steven de Jong (UM)
Fairness in Multi-Agent Systems

2009-14 Maksym Korotkiy (VU)

From ontology-enabled services to service-enabled ontologies (making ontologies work in e-science with ONTO-SOA)

2009-15 Rinke Hoekstra (UVA)
Ontology Representation - Design Patterns and Ontologies that Make Sense

2009-16 Fritz Reul (UvT)
New Architectures in Computer Chess

2009-17 Laurens van der Maaten (UvT)
Feature Extraction from Visual Data

2009-18 Fabian Groffen (CWI)
Armada, An Evolving Database System

2009-19 Valentin Robu (CWI)
Modeling Preferences, Strategic Reasoning and Collaboration in Agent-Mediated Electronic Markets

2009-20 Bob van der Vecht (UU)
Adjustable Autonomy: Controling Influences on Decision Making

2009-21 Stijn Vanderlooy (UM)
Ranking and Reliable Classification

2009-22 Pavel Serdyukov (UT)
Search For Expertise: Going beyond direct evidence

2009-23 Peter Hofgesang (VU)
Modelling Web Usage in a Changing Environment

2009-24 Annerieke Heuvelink (VUA)
Cognitive Models for Training Simulations

2009-25 Alex van Ballegooij (CWI)
"RAM: Array Database Management through Relational Mapping"

2009-26 Fernando Koch (UU)
An Agent-Based Model for the Development of Intelligent Mobile Services

2009-27 Christian Glahn (OU)
Contextual Support of social Engagement and Reflection on the Web

2009-28 Sander Evers (UT)
Sensor Data Management with Probabilistic Models

2009-29 Stanislav Pokraev (UT)
Model-Driven Semantic Integration of Service-Oriented Applications

2009-30 Marcin Zukowski (CWI)

Balancing vectorized query execution with bandwidth-optimized storage

2009-31 Sofiya Katrenko (UVA)
A Closer Look at Learning Relations from Text

2009-32 Rik Farenhorst (VU) and Remco de Boer (VU)
Architectural Knowledge Management: Supporting Architects and Auditors

2009-33 Khiet Truong (UT)
How Does Real Affect Affect Affect Recognition In Speech?

2009-34 Inge van de Weerd (UU)
Advancing in Software Product Management: An Incremental Method Engineering Approach

2009-35 Wouter Koelewijn (UL)
Privacy en Politiegegevens; Over geautomatiseerde normatieve informatie-uitwisseling

2009-36 Marco Kalz (OUN)
Placement Support for Learners in Learning Networks

2009-37 Hendrik Drachsler (OUN)
Navigation Support for Learners in Informal Learning Networks

2009-38 Riina Vuorikari (OU)
Tags and self-organisation: a metadata ecology for learning resources in a multilingual context

2009-39 Christian Stahl (TUE, Humboldt-Universitaet zu Berlin)
Service Substitution – A Behavioral Approach Based on Petri Nets

2009-40 Stephan Raaijmakers (UvT)
Multinomial Language Learning: Investigations into the Geometry of Language

2009-41 Igor Berezhnyy (UvT)
Digital Analysis of Paintings

2009-42 Toine Bogers (UvT)
Recommender Systems for Social Bookmarking

2009-43 Virginia Nunes Leal Franqueira (UT)
Finding Multi-step Attacks in Computer Networks using Heuristic Search and Mobile Ambients

2009-44 Roberto Santana Tapia (UT)
Assessing Business-IT Alignment in Networked Organizations

2009-45 Jilles Vreeken (UU)
Making Pattern Mining Useful

2009-46 Loredana Afanasiev (UvA)
Querying XML: Benchmarks and Recursion

2010

2010-01 Matthijs van Leeuwen (UU)
Patterns that Matter

2010-02 Ingo Wassink (UT)
Work flows in Life Science

2010-03 Joost Geurts (CWI)
A Document Engineering Model and Processing Framework for Multimedia documents

2010-04 Olga Kulyk (UT)
Do You Know What I Know? Situational Awareness of Co-located Teams in Multidisplay Environments

2010-05 Claudia Hauff (UT)
Predicting the Effectiveness of Queries and Retrieval Systems

2010-06 Sander Bakkes (UvT)
Rapid Adaptation of Video Game AI

2010-07 Wim Fikkert (UT)
Gesture interaction at a Distance

2010-08 Krzysztof Siewicz (UL)
Towards an Improved Regulatory Framework of Free Software. Protecting user freedoms in a world of software communities and eGovernments

2010-09 Hugo Kielman (UL)
A Politiele gegevensverwerking en Privacy, Naar een effectieve waarborging

2010-10 Rebecca Ong (UL)
Mobile Communication and Protection of Children

2010-11 Adriaan Ter Mors (TUD)
The world according to MARP: Multi-Agent Route Planning

2010-12 Susan van den Braak (UU)
Sensemaking software for crime analysis

2010-13 Gianluigi Folino (RUN)
High Performance Data Mining using Bio-inspired techniques

2010-14 Sander van Splunter (VU)
Automated Web Service Reconfiguration

2010-15 Lianne Bodenstaff (UT)
Managing Dependency Relations in Inter-Organizational Models

2010-16 Sicco Verwer (TUD)
Efficient Identification of Timed Automata, theory and practice

2010-17 Spyros Kotoulas (VU)
Scalable Discovery of Networked Resources: Algorithms, Infrastructure, Applications

2010-18 Charlotte Gerritsen (VU)
Caught in the Act: Investigating Crime by Agent-Based Simulation

2010-19 Henriette Cramer (UvA)
People's Responses to Autonomous and Adaptive Systems

2010-20 Ivo Swartjes (UT)
Whose Story Is It Anyway? How Improv Informs Agency and Authorship of Emergent Narrative

2010-21 Harold van Heerde (UT)
Privacy-aware data management by means of data degradation

2010-22 Michiel Hildebrand (CWI)
End-user Support for Access to Heterogeneous Linked Data

2010-23 Bas Steunebrink (UU)
The Logical Structure of Emotions

2010-24 Dmytro Tykhonov
Designing Generic and Efficient Negotiation Strategies

2010-25 Zulfiqar Ali Memon (VU)
Modelling Human-Awareness for Ambient Agents: A Human Mindreading Perspective

2010-26 Ying Zhang (CWI)
XRPC: Efficient Distributed Query Processing on Heterogeneous XQuery Engines

2010-27 Marten Voulon (UL)
Automatisch contracteren

2010-28 Arne Koopman (UU)
Characteristic Relational Patterns

2010-29 Stratos Idreos(CWI)

Database Cracking: Towards Auto-tuning Database Kernels

2010-30 Marieke van Erp (UvT)
Accessing Natural History - Discoveries in data cleaning, structuring, and retrieval

2010-31 Victor de Boer (UVA)
Ontology Enrichment from Heterogeneous Sources on the Web

2010-32 Marcel Hiel (UvT)
An Adaptive Service Oriented Architecture: Automatically solving Interoperability Problems

2010-33 Robin Aly (UT)
Modeling Representation Uncertainty in Concept-Based Multimedia Retrieval

2010-34 Teduh Dirgahayu (UT)
Interaction Design in Service Compositions

2010-35 Dolf Trieschnigg (UT)
Proof of Concept: Concept-based Biomedical Information Retrieval

2010-36 Jose Janssen (OU)
Paving the Way for Lifelong Learning; Facilitating competence development through a learning path specification

2010-37 Niels Lohmann (TUE)
Correctness of services and their composition

2010-38 Dirk Fahland (TUE)
From Scenarios to components

2010-39 Ghazanfar Farooq Siddiqui (VU)
Integrative modeling of emotions in virtual agents

2010-40 Mark van Assem (VU)
Converting and Integrating Vocabularies for the Semantic Web

2010-41 Guillaume Chaslot (UM)
Monte-Carlo Tree Search

2010-42 Sybren de Kinderen (VU)
Needs-driven service bundling in a multi-supplier setting - the computational e3-service approach

2010-43 Peter van Kranenburg (UU)
A Computational Approach to Content-Based Retrieval of Folk Song Melodies

2010-44 Pieter Bellekens (TUE)

An Approach towards Context-sensitive and User-adapted Access to Heterogeneous Data Sources, Illustrated in the Television Domain

2010-45 Vasilios Andrikopoulos (UvT)
A theory and model for the evolution of software services

2010-46 Vincent Pijpers (VU)
e3alignment: Exploring Inter-Organizational Business-ICT Alignment

2010-47 Chen Li (UT)
Mining Process Model Variants: Challenges, Techniques, Examples

2010-48 Withdrawn

2010-49 Jahn-Takeshi Saito (UM)
Solving difficult game positions

2010-50 Bouke Huurnink (UVA)
Search in Audiovisual Broadcast Archives

2010-51 Alia Khairia Amin (CWI)
Understanding and supporting information seeking tasks in multiple sources

2010-52 Peter-Paul van Maanen (VU)
Adaptive Support for Human-Computer Teams: Exploring the Use of Cognitive Models of Trust and Attention

2010-53 Edgar Meij (UVA)
Combining Concepts and Language Models for Information Access

2011

2011-01 Botond Cseke (RUN)
Variational Algorithms for Bayesian Inference in Latent Gaussian Models

2011-02 Nick Tinnemeier(UU)
Organizing Agent Organizations. Syntax and Operational Semantics of an Organization-Oriented Programming Language

2011-03 Jan Martijn van der Werf (TUE)
Compositional Design and Verification of Component-Based Information Systems

2011-04 Hado van Hasselt (UU)
Insights in Reinforcement Learning; Formal analysis and empirical evaluation of temporal-difference learning algorithms

2011-05 Base van der Raadt (VU)
Enterprise Architecture Coming of Age - Increasing the Performance of an Emerging Discipline.

2011-06 Yiwen Wang (TUE)
Semantically-Enhanced Recommendations in Cultural Heritage

2011-07 Yujia Cao (UT)
Multimodal Information Presentation for High Load Human Computer Interaction

2011-08 Nieske Vergunst (UU)
BDI-based Generation of Robust Task-Oriented Dialogues

2011-09 Tim de Jong (OU)
Contextualised Mobile Media for Learning

2011-10 Bart Bogaert (UvT)
Cloud Content Contention

2011-11 Dhaval Vyas (UT)
Designing for Awareness: An Experience-focused HCI Perspective

2011-12 Carmen Bratosin (TUE)
Grid Architecture for Distributed Process Mining

2011-13 Xiaoyu Mao (UvT)
Airport under Control. Multiagent Scheduling for Airport Ground Handling

2011-14 Milan Lovric (EUR)
Behavioral Finance and Agent-Based Artificial Markets

2011-15 Marijn Koolen (UvA)
The Meaning of Structure: the Value of Link Evidence for Information Retrieval

2011-16 Maarten Schadd (UM)
Selective Search in Games of Different Complexity

2011-17 Jiyin He (UVA)
Exploring Topic Structure: Coherence, Diversity and Relatedness

2011-18 Mark Ponsen (UM)
Strategic Decision-Making in complex games

2011-19 Ellen Rusman (OU)
The Mind ' s Eye on Personal Profiles

2011-20 Qing Gu (VU)
Guiding service-oriented software engineering - A view-based approach

2011-21 Linda Terlouw (TUD)
Modularization and Specification of Service-Oriented Systems

2011-22 Junte Zhang (UVA)
System Evaluation of Archival Description and Access

2011-23 Wouter Weerkamp (UVA)
Finding People and their Utterances in Social Media

2011-24 Herwin van Welbergen (UT)
Behavior Generation for Interpersonal Coordination with Virtual Humans On Specifying, Scheduling and Realizing Multimodal Virtual Human Behavior

2011-25 Syed Waqar ul Qounain Jaffry (VU))
Analysis and Validation of Models for Trust Dynamics

2011-26 Matthijs Aart Pontier (VU)
Virtual Agents for Human Communication - Emotion Regulation and Involvement-Distance Trade-Offs in Embodied Conversational Agents and Robots

2011-27 Aniel Bhulai (VU)
Dynamic website optimization through autonomous management of design patterns

2011-28 Rianne Kaptein(UVA)
Effective Focused Retrieval by Exploiting Query Context and Document Structure

2011-29 Faisal Kamiran (TUE)
Discrimination-aware Classification

2011-30 Egon van den Broek (UT)
Affective Signal Processing (ASP): Unraveling the mystery of emotions

2011-31 Ludo Waltman (EUR)
Computational and Game-Theoretic Approaches for Modeling Bounded Rationality

2011-32 Nees-Jan van Eck (EUR)
Methodological Advances in Bibliometric Mapping of Science

2011-33 Tom van der Weide (UU)
Arguing to Motivate Decisions

2011-34 Paolo Turrini (UU)
Strategic Reasoning in Interdependence: Logical and Game-theoretical Investigations

2011-35 Maaike Harbers (UU)
Explaining Agent Behavior in Virtual Training

2011-36 Erik van der Spek (UU)
Experiments in serious game design: a cognitive approach

2011-37 Adriana Burlutiu (RUN)
Machine Learning for Pairwise Data, Applications for Preference Learning and Supervised Network Inference

2011-38 Nyree Lemmens (UM)
Bee-inspired Distributed Optimization

2011-39 Joost Westra (UU)
Organizing Adaptation using Agents in Serious Games

2011-40 Viktor Clerc (VU)
Architectural Knowledge Management in Global Software Development

2011-41 Luan Ibraimi (UT)
Cryptographically Enforced Distributed Data Access Control

2011-42 Michal Sindlar (UU)
Explaining Behavior through Mental State Attribution

2011-43 Henk van der Schuur (UU)
Process Improvement through Software Operation Knowledge

2011-44 Boris Reuderink (UT)
Robust Brain-Computer Interfaces

2011-45 Herman Stehouwer (UvT)
Statistical Language Models for Alternative Sequence Selection

2011-46 Beibei Hu (TUD)
Towards Contextualized Information Delivery: A Rule-based Architecture for the Domain of Mobile Police Work

2011-47 Azizi Bin Ab Aziz(VU)
Exploring Computational Models for Intelligent Support of Persons with Depression

2011-48 Mark Ter Maat (UT)
Response Selection and Turn-taking for a Sensitive Artificial Listening Agent

2011-49 Andreea Niculescu (UT)
Conversational interfaces for task-oriented spoken dialogues: design aspects influencing interaction quality

2012

2012-01 Terry Kakeeto (UvT)
Relationship Marketing for SMEs in Uganda

2012-02 Muhammad Umair(VU)
Adaptivity, emotion, and Rationality in Human and Ambient Agent Models

2012-03 Adam Vanya (VU)
Supporting Architecture Evolution by Mining Software Repositories

2012-04 Jurriaan Souer (UU)
Development of Content Management System-based Web Applications

2012-05 Marijn Plomp (UU)
Maturing Interorganisational Information Systems

2012-06 Wolfgang Reinhardt (OU)
Awareness Support for Knowledge Workers in Research Networks

2012-07 Rianne van Lambalgen (VU)
When the Going Gets Tough: Exploring Agent-based Models of Human Performance under Demanding Conditions

2012-08 Gerben de Vries (UVA)
Kernel Methods for Vessel Trajectories

2012-09 Ricardo Neisse (UT)
Trust and Privacy Management Support for Context-Aware Service Platforms

2012-10 David Smits (TUE)
Towards a Generic Distributed Adaptive Hypermedia Environment

2012-11 J.C.B. Rantham Prabhakara (TUE)
Process Mining in the Large: Preprocessing, Discovery, and Diagnostics

2012-12 Kees van der Sluijs (TUE)
Model Driven Design and Data Integration in Semantic Web Information Systems

2012-13 Suleman Shahid (UvT)
Fun and Face: Exploring non-verbal expressions of emotion during playful interactions

2012-14 Evgeny Knutov(TUE)
Generic Adaptation Framework for Unifying Adaptive Web-based Systems

2012-15 Natalie van der Wal (VU)
Social Agents. Agent-Based Modelling of Integrated Internal and Social Dynamics of Cognitive and Affective Processes.

2012-16 Fiemke Both (VU)
Helping people by understanding them - Ambient Agents supporting task execution and depression treatment

2012-17 Amal Elgammal (UvT)
Towards a Comprehensive Framework for Business Process Compliance

2012-18 Eltjo Poort (VU)
Improving Solution Architecting Practices

2012-19 Helen Schonenberg (TUE)
What's Next? Operational Support for Business Process Execution

2012-20 Ali Bahramisharif (RUN)
Covert Visual Spatial Attention, a Robust Paradigm for Brain-Computer Interfacing

2012-21 Roberto Cornacchia (TUD)
Querying Sparse Matrices for Information Retrieval

2012-22 Thijs Vis (UvT)
Intelligence, politie en veiligheidsdienst: verenigbare grootheden?

2012-23 Christian Muehl (UT)
Toward Affective Brain-Computer Interfaces: Exploring the Neurophysiology of Affect during Human Media Interaction

2012-24 Laurens van der Werff (UT)
Evaluation of Noisy Transcripts for Spoken Document Retrieval

2012-25 Silja Eckartz (UT)
Managing the Business Case Development in Inter-Organizational IT Projects: A Methodology and its Application

2012-26 Emile de Maat (UVA)
Making Sense of Legal Text

2012-27 Hayrettin Gurkok (UT)
Mind the Sheep! User Experience Evaluation & Brain-Computer Interface Games

2012-28 Nancy Pascall (UvT)
Engendering Technology Empowering Women

2012-29 Almer Tigelaar (UT)
Peer-to-Peer Information Retrieval

2012-30 Alina Pommeranz (TUD)
Designing Human-Centered Systems for Reflective Decision Making

2012-31 Emily Bagarukayo (RUN)
A Learning by Construction Approach for Higher Order Cognitive Skills Improvement, Building Capacity and Infrastructure

2012-32 Wietske Visser (TUD)
Qualitative multi-criteria preference representation and reasoning

2012-33 Rory Sie (OUN)
Coalitions in Cooperation Networks (COCOON)

2012-34 Pavol Jancura (RUN)
Evolutionary analysis in PPI networks and applications

2012-35 Evert Haasdijk (VU)
Never Too Old To Learn – On-line Evolution of Controllers in Swarm- and Modular Robotics

2012-36 Denis Ssebugwawo (RUN)
Analysis and Evaluation of Collaborative Modeling Processes

2012-37 Agnes Nakakawa (RUN)
A Collaboration Process for Enterprise Architecture Creation

2012-38 Selmar Smit (VU)
Parameter Tuning and Scientific Testing in Evolutionary Algorithms

2012-39 Hassan Fatemi (UT)
Risk-aware design of value and coordination networks

2012-40 Agus Gunawan (UvT)
Information Access for SMEs in Indonesia

2012-41 Sebastian Kelle (OU)
Game Design Patterns for Learning

2012-42 Dominique Verpoorten (OU)
Reflection Amplifiers in self-regulated Learning

2012-43 Withdrawn

2012-44 Anna Tordai (VU)
On Combining Alignment Techniques

2012-45 Benedikt Kratz (UvT)
A Model and Language for Business-aware Transactions

2012-46 Simon Carter (UVA)
Exploration and Exploitation of Multilingual Data for Statistical Machine Translation

2012-47 Manos Tsagkias (UVA)
Mining Social Media: Tracking Content and Predicting Behavior

2012-48 Jorn Bakker (TUE)
Handling Abrupt Changes in Evolving Time-series Data

2012-49 Michael Kaisers (UM)
Learning against Learning - Evolutionary dynamics of reinforcement learning algorithms in strategic interactions

2012-50 Steven van Kervel (TUD)
Ontologogy driven Enterprise Information Systems Engineering

2012-51 Jeroen de Jong (TUD)
Heuristics in Dynamic Sceduling; a practical framework with a case study in elevator dispatching

2013

2013-01 Viorel Milea (EUR)
News Analytics for Financial Decision Support

2013-02 Erietta Liarou (CWI)
MonetDB/DataCell: Leveraging the Column-store Database Technology for Efficient and Scalable Stream Processing

2013-03 Szymon Klarman (VU)
Reasoning with Contexts in Description Logics

2013-04 Chetan Yadati(TUD)
Coordinating autonomous planning and scheduling

2013-05 Dulce Pumareja (UT)
Groupware Requirements Evolutions Patterns

2013-06 Romulo Goncalves(CWI)
The Data Cyclotron: Juggling Data and Queries for a Data Warehouse Audience

2013-07 Giel van Lankveld (UvT)
Quantifying Individual Player Differences

2013-08 Robbert-Jan Merk(VU)
Making enemies: cognitive modeling for opponent agents in fighter pilot simulators

2013-09 Fabio Gori (RUN)
Metagenomic Data Analysis: Computational Methods and Applications

2013-10 Jeewanie Jayasinghe Arachchige(UvT)
A Unified Modeling Framework for Service Design.

2013-11 Evangelos Pournaras(TUD)
Multi-level Reconfigurable Self-organization in Overlay Services

2013-12 Marian Razavian(VU)
Knowledge-driven Migration to Services

2013-13 Mohammad Safiri(UT)
Service Tailoring: User-centric creation of integrated IT-based homecare services to support independent living of elderly

2013-14 Jafar Tanha (UVA)
Ensemble Approaches to Semi-Supervised Learning Learning

2013-15 Daniel Hennes (UM)
Multiagent Learning - Dynamic Games and Applications

2013-16 Eric Kok (UU)
Exploring the practical benefits of argumentation in multi-agent deliberation

2013-17 Koen Kok (VU)
The PowerMatcher: Smart Coordination for the Smart Electricity Grid

2013-18 Jeroen Janssens (UvT)
Outlier Selection and One-Class Classification

2013-19 Renze Steenhuizen (TUD)
Coordinated Multi-Agent Planning and Scheduling

2013-20 Katja Hofmann (UvA)
Fast and Reliable Online Learning to Rank for Information Retrieval

2013-21 Sander Wubben (UvT)
Text-to-text generation by monolingual machine translation

2013-22 Tom Claassen (RUN)
Causal Discovery and Logic

2013-23 Patricio de Alencar Silva(UvT)
Value Activity Monitoring

2013-24 Haitham Bou Ammar (UM)
Automated Transfer in Reinforcement Learning

2013-25 Agnieszka Anna Latoszek-Berendsen (UM)

Intention-based Decision Support. A new way of representing and implementing clinical guidelines in a Decision Support System

2013-26 Alireza Zarghami (UT)
Architectural Support for Dynamic Homecare Service Provisioning

2013-27 Mohammad Huq (UT)
Inference-based Framework Managing Data Provenance

2013-28 Frans van der Sluis (UT)
When Complexity becomes Interesting: An Inquiry into the Information eXperience

2013-29 Iwan de Kok (UT)
Listening Heads

2013-30 Joyce Nakatumba (TUE)
Resource-Aware Business Process Management: Analysis and Support

2013-31 Dinh Khoa Nguyen (UvT)
Blueprint Model and Language for Engineering Cloud Applications

2013-32 Kamakshi Rajagopal (OUN)
Networking For Learning; The role of Networking in a Lifelong Learner's Professional Development

2013-33 Qi Gao (TUD)
User Modeling and Personalization in the Microblogging Sphere

2013-34 Kien Tjin-Kam-Jet (UT)
Distributed Deep Web Search

2013-35 Abdallah El Ali (UvA)
Minimal Mobile Human Computer Interaction

2013-36 Than Lam Hoang (TUe)
Pattern Mining in Data Streams

2013-37 Dirk Börner (OUN)
Ambient Learning Displays

2013-38 Eelco den Heijer (VU)
Autonomous Evolutionary Art

2013-39 Joop de Jong (TUD)
A Method for Enterprise Ontology based Design of Enterprise Information Systems

2013-40 Pim Nijssen (UM)
Monte-Carlo Tree Search for Multi-Player Games

2013-41 Jochem Liem (UVA)
Supporting the Conceptual Modelling of Dynamic Systems: A Knowledge Engineering Perspective on Qualitative Reasoning

2013-42 Léon Planken (TUD)
Algorithms for Simple Temporal Reasoning

2013-43 Marc Bron (UVA)
Exploration and Contextualization through Interaction and Concepts

2014

2014-01 Nicola Barile (UU)
Studies in Learning Monotone Models from Data

2014-02 Fiona Tuliyano (RUN)
Combining System Dynamics with a Domain Modeling Method

2014-03 Sergio Raul Duarte Torres (UT)
Information Retrieval for Children: Search Behavior and Solutions

2014-04 Hanna Jochmann-Mannak (UT)
Websites for children: search strategies and interface design - Three studies on children's search performance and evaluation

2014-05 Jurriaan van Reijsen (UU)
Knowledge Perspectives on Advancing Dynamic Capability

2014-06 Damian Tamburri (VU)
Supporting Networked Software Development

2014-07 Arya Adriansyah (TUE)
Aligning Observed and Modeled Behavior

2014-08 Samur Araujo (TUD)
Data Integration over Distributed and Heterogeneous Data Endpoints

2014-09 Philip Jackson (UvT)
Toward Human-Level Artificial Intelligence: Representation and Computation of Meaning in Natural Language

2014-10 Ivan Salvador Razo Zapata (VU)
Service Value Networks

2014-11 Janneke van der Zwaan (TUD)
An Empathic Virtual Buddy for Social Support

2014-12 Willem van Willigen (VU)
Look Ma, No Hands: Aspects of Autonomous Vehicle Control

2014-13 Arlette van Wissen (VU)
Agent-Based Support for Behavior Change: Models and Applications in Health and Safety Domains

2014-14 Yangyang Shi (TUD)
Language Models With Meta-information

2014-15 Natalya Mogles (VU)
Agent-Based Analysis and Support of Human Functioning in Complex Socio-Technical Systems: Applications in Safety and Healthcare

2014-16 Krystyna Milian (VU)
Supporting trial recruitment and design by automatically interpreting eligibility criteria

2014-17 Kathrin Dentler (VU)
Computing healthcare quality indicators automatically: Secondary Use of Patient Data and Semantic Interoperability

2014-18 Mattijs Ghijsen (VU)
Methods and Models for the Design and Study of Dynamic Agent Organizations

2014-19 Vinicius Ramos (TUE)
Adaptive Hypermedia Courses: Qualitative and Quantitative Evaluation and Tool Support

2014-20 Mena Habib (UT)
Named Entity Extraction and Disambiguation for Informal Text: The Missing Link

2014-21 Kassidy Clark (TUD)
Negotiation and Monitoring in Open Environments

2014-22 Marieke Peeters (UU)
Personalized Educational Games - Developing agent-supported scenario-based training

2014-23 Eleftherios Sidirourgos (UvA/CWI)
Space Efficient Indexes for the Big Data Era

2014-24 Davide Ceolin (VU)
Trusting Semi-structured Web Data

2014-25 Martijn Lappenschaar (RUN)
New network models for the analysis of disease interaction

2014-26 Tim Baarslag (TUD)

What to Bid and When to Stop

2014-27 Rui Jorge Almeida (EUR)
Conditional Density Models Integrating Fuzzy and Probabilistic Representations of Uncertainty

2014-28 Anna Chmielowiec (VU)
Decentralized k-Clique Matching

2014-29 Jaap Kabbedijk (UU)
Variability in Multi-Tenant Enterprise Software

2014-30 Peter de Cock (UvT)
Anticipating Criminal Behaviour

2014-31 Leo van Moergestel (UU)
Agent Technology in Agile Multiparallel Manufacturing and Product Support

2014-32 Naser Ayat (UvA)
On Entity Resolution in Probabilistic Data

2014-33 Tesfa Tegegne (RUN)
Service Discovery in eHealth

2014-34 Christina Manteli(VU)
The Effect of Governance in Global Software Development: Analyzing Transactive Memory Systems.

2014-35 Joost van Ooijen (UU)
Cognitive Agents in Virtual Worlds: A Middleware Design Approach

2014-36 Joos Buijs (TUE)
Flexible Evolutionary Algorithms for Mining Structured Process Models

2014-37 Maral Dadvar (UT)
Experts and Machines United Against Cyberbullying

2014-38 Danny Plass-Oude Bos (UT)
Making brain-computer interfaces better: improving usability through post-processing.

2014-39 Jasmina Maric (UvT)
Web Communities, Immigration, and Social Capital

2014-40 Walter Omona (RUN)
A Framework for Knowledge Management Using ICT in Higher Education

2014-41 Frederic Hogenboom (EUR)
Automated Detection of Financial Events in News Text

2014-42 Carsten Eijckhof (CWI/TUD)
Contextual Multidimensional Relevance Models

2014-43 Kevin Vlaanderen (UU)
Supporting Process Improvement using Method Increments

2014-44 Paulien Meesters (UvT)
Intelligent Blauw. Met als ondertitel: Intelligence-gestuurde politiezorg in gebiedsgebonden eenheden.

2014-45 Birgit Schmitz (OUN)
Mobile Games for Learning: A Pattern-Based Approach

2014-46 Ke Tao (TUD)
Social Web Data Analytics: Relevance, Redundancy, Diversity

2014-47 Shangsong Liang (UVA)
Fusion and Diversification in Information Retrieval

2015

2015-01 Niels Netten (UvA)
Machine Learning for Relevance of Information in Crisis Response

2015-02 Faiza Bukhsh (UvT)
Smart auditing: Innovative Compliance Checking in Customs Controls

2015-03 Twan van Laarhoven (RUN)
Machine learning for network data

2015-04 Howard Spoelstra (OUN)
Collaborations in Open Learning Environments

2015-05 Christoph Bösch(UT)
Cryptographically Enforced Search Pattern Hiding

2015-06 Farideh Heidari (TUD)
Business Process Quality Computation - Computing Non-Functional Requirements to Improve Business Processes

2015-07 Maria-Hendrike Peetz(UvA)
Time-Aware Online Reputation Analysis

2015-08 Jie Jiang (TUD)
Organizational Compliance: An agent-based model for designing and evaluating organizational interactions

2015-09 Randy Klaassen(UT)
HCI Perspectives on Behavior Change Support Systems

2015-10 Henry Hermans (OUN)
OpenU: design of an integrated system to support lifelong learning

2015-11 Yongming Luo(TUE)
Designing algorithms for big graph datasets: A study of computing bisimulation and joins

2015-12 Julie M. Birkholz (VU)
Modi Operandi of Social Network Dynamics: The Effect of Context on Scientific Collaboration Networks

2015-13 Giuseppe Procaccianti(VU)
Energy-Efficient Software

2015-14 Bart van Straalen (UT)
A cognitive approach to modeling bad news conversations

2015-15 Klaas Andries de Graaf (VU)
Ontology-based Software Architecture Documentation

2015-16 Changyun Wei (UT)
Cognitive Coordination for Cooperative Multi-Robot Teamwork

2015-17 André van Cleeff (UT)
Physical and Digital Security Mechanisms: Properties, Combinations and Trade-offs

2015-18 Holger Pirk (CWI)
Waste Not, Want Not! - Managing Relational Data in Asymmetric Memories

2015-19 Bernardo Tabuenca (OUN)
Ubiquitous Technology for Lifelong Learners

2015-20 Loïs Vanhée(UU)
Using Culture and Values to Support Flexible Coordination

2015-21 Sibren Fetter (OUN)
Using Peer-Support to Expand and Stabilize Online Learning

015-22 Zhemin Zhu(UT)
Co-occurrence Rate Networks

2015-23 Luit Gazendam (VU)
Cataloguer Support in Cultural Heritage

2015-24 Richard Berendsen (UVA)
Finding People, Papers, and Posts: Vertical Search Algorithms and Evaluation

2015-25 Steven Woudenberg (UU)
Bayesian Tools for Early Disease Detection

2015-26 Alexander Hogenboom (EUR)
Sentiment Analysis of Text Guided by Semantics and Structure

2015-27 Sándor Héman (CWI)
Updating compressed column-stores

2015-28 Janet Bagorogoza (TiU)
Knowledge Management and High Performance; The Uganda Financial Institutions Model for HPO

2015-29 Hendrik Baier (UM)
Monte-Carlo Tree Search Enhancements for One-Player and Two-Player Domains

2015-30 Kiavash Bahreini (OUN)
Real-time Multimodal Emotion Recognition in E-Learning

2015-31 Yakup Koç (TUD)
On Robustness of Power Grids

2015-32 Jerome Gard (UL)
Corporate Venture Management in SMEs

2015-33 Frederik Schadd (UM)
Ontology Mapping with Auxiliary Resources

2015-34 Victor de Graaff (UT)
Geosocial Recommender Systems

2015-35 Junchao Xu (TUD)
Affective Body Language of Humanoid Robots: Perception and Effects in Human Robot Interaction

2016

2016-01 Syed Saiden Abbas (RUN)
Recognition of Shapes by Humans and Machines

2016-02 Michiel Christiaan Meulendijk (UU)
Optimizing medication reviews through decision support: prescribing a better pill to swallow

2016-03 Maya Sappelli (RUN)
Knowledge Work in Context: User Centered Knowledge Worker Support

2016-04 Laurens Rietveld (VU)
Publishing and Consuming Linked Data

2016-05 Evgeny Sherkhonov (UVA)
Expanded Acyclic Queries: Containment and an Application in Explaining Missing Answers

2016-06 Michel Wilson (TUD)
Robust scheduling in an uncertain environment

2016-07 Jeroen de Man (VU)
Measuring and modeling negative emotions for virtual training

2016-08 Matje van de Camp (TiU)
A Link to the Past: Constructing Historical Social Networks from Unstructured Data

2016-09 Archana Nottamkandath (VU)
Trusting Crowdsourced Information on Cultural Artefacts

2016-10 George Karafotias (VUA)
Parameter Control for Evolutionary Algorithms

2016-11 Anne Schuth (UVA)
Search Engines that Learn from Their Users

2016-12 Max Knobbout (UU)
Logics for Modelling and Verifying Normative Multi-Agent Systems

2016-13 Nana Baah Gyan (VU)
The Web, Speech Technologies and Rural Development in West Africa - An ICT4D Approach

2016-14 Ravi Khadka (UU)
Revisiting Legacy Software System Modernization

SUMMARY

Building systems that can automatically reason about situations and take decisions, in spite of uncertainty about such situations, is an important goal in *artificial intelligence* (AI). *Probabilistic logics* aim at combining the properties of logic, that is they provide a structured way of expressing knowledge and a mechanical way of reasoning about such knowledge, with the ability of *probability theory* to deal with uncertainty. Such probabilistic logics can serve as a basis to automate uncertainty reasoning, based on a structured and interpretable representation of knowledge.

There is a wide spectrum of probabilistic logic languages, which differ in the fundamental balance between how expressive a language is and how hard it is to reason about knowledge expressed in the language. On the one hand, a language should be expressive enough to allow one to express all knowledge required to model a problem at hand. On the other hand, it should still be possible to draw useful conclusions in reasonable time.

In this thesis we propose probabilistic logics with a unique balance between expressiveness and hardness of reasoning, which perfectly matches the requirements for many problem domains. On the expressivity side, we want to support hybrid models, which means that not only discrete factors, such as whether it is sunny, rainy or snowing, can be incorporated, but also continuous ones, such as the temperature. The ability to support hybrid models is essential to express virtually any non-trivial knowledge about the physical world. On the reasoning side, we want to preserve *soundness* of reasoning, which means that conclusions drawn from knowledge agree with that knowledge. This may seem a basic requirement, but all common *inference* methods for hybrid models are either restricted to only a small class of such models, or provide only unsound reasoning. The latter are based on approximations, which usually provide good results, but that come without any guarantees. This is problematic for domains in which wrong decisions may have a huge impact.

This thesis covers a wide range of results. We provide a theoretical basis for probabilistic logics with the properties outlined above, but we also consider the problem of how much time reasoning takes. We propose a way to structure probabilistic knowledge that allows for efficient reasoning, comparable to common, less expressive probabilistic logic languages. This is approached both theoretically and also practically, illustrated by concrete reasoning algorithms, which we evaluate experimentally. We also design practical languages, suitable for knowledge representation, and show applicability using a challenging real-world problem. This was done in the context of an industrial project, in which we built a model to reason about malicious behaviour of vessels.

SAMENVATTING

Het ontwikkelen van systemen die automatisch over situaties kunnen redeneren en daarbij beslissingen kunnen nemen, ondanks de onzekerheid over die situaties, is een belangrijk doel van de *kunstmatige intelligentie* (KI). *Probabilistische logica's* combineren eigenschappen van logica, in het bijzonder om kennis op een gestructureerde manier uit te kunnen drukken en ermee op mechanische wijze te kunnen redeneren, met de mogelijkheden van kansrekening om met onzekerheid om te gaan. Probabilistische logica's kunnen daarom dienen als een basis voor het automatisch redeneren met onzekerheid gegeven een gestructureerde en interpreteerbare representatie van de kennis.

Er is een breed spectrum aan probabilistische logische talen; zij verschillen in de fundamentele balans tussen expressiviteit en de moeilijkheid om over kennis, uitgedrukt in de taal, efficiënt te kunnen redeneren. Enerzijds moet een taal voldoende expressief zijn om alle voor een probleem relevante kennis uit te kunnen drukken. Anderzijds moet het mogelijk zijn om in redelijke tijd de computer een conclusie te laten trekken.

In dit proefschrift worden probabilistische logica's geïntroduceerd die een unieke balans bezitten tussen expressiviteit en de computationele moeilijkheid van redeneren. De gekozen balans past goed bij de eisen van veel probleemdomeinen. Wat betreft de expressiviteit worden hybride modellen ondersteund, wat wil zeggen dat niet alleen maar discrete variabelen, zoals of het zonnig is, regent of sneeuwt, kunnen worden gemodelleerd, maar ook continue variabelen, zoals de omgevingstemperatuur. Deze ondersteuning voor hybride modellen is essentieel om kennis over de fysieke wereld op een natuurlijke manier uit te kunnen drukken. Er wordt een vorm van redeneren ondersteund die de *correctheid* (gezondheid) van de redeneren garandeert. Dit wil zeggen dat conclusies die worden getrokken op basis van kennis uit deze kennis volgen. Dit lijkt misschien een voor de hand liggende eis, maar alle veelgebruikte inferentiemethodes voor hybride modellen zijn beperkt tot een kleine klasse van dit soort modellen of ondersteunen slechts een niet-gezonde manier van redeneren. Er wordt dan gebruik gemaakt van benaderingen, die geen garantie voor nauwkeurigheid opleveren. Dit is problematisch in domeinen waar foute beslissingen grote consequenties kunnen hebben.

Dit proefschrift bevat een diversiteit aan resultaten. Eerst wordt een theoretische basis ontwikkeld voor probabilistische logica's met eigenschappen zoals hierboven besproken. Ook wordt aandacht gegeven aan het probleem van de hoeveelheid computationele tijd dat redeneren nodig heeft. Een methode wordt ontwikkeld om probabilistische kennis zodanig te structureren dat efficiënt redeneren mogelijk gemaakt wordt, vergelijkbaar met veel gebruikte, minder expressieve probabilistische logica's. Dit wordt zowel theoretisch als ook praktisch geïllustreerd aan hand van redeneeralgoritmen die ook experimenteel worden

geëvalueerd. Daarnaast worden ook praktische talen voor kennisrepresentatie ontwikkeld. Aan de hand van een uitdagend realistisch probleem wordt aangetoond dat deze talen ook praktisch bruikbaar zijn. Ook de industriële toepasbaarheid is met een computermodel, dat ondersteuning biedt om over verdacht gedrag van schepen te redeneren, aangetoond.

ACKNOWLEDGMENTS

I could only write this thesis thanks to the help and support of many people. First of all, I owe a great debt of gratitude to my promoter Peter for his invaluable support. Your critical feedback helped to enormously improve my work. Thank you for your guidance, but also the freedom to pursue my own ideas. Also I want to especially thank my copromotor Marina. I really enjoyed our collaboration. Working on a project, such as the *METIS* project, together with industry and various academic partners, can sometimes feel like being confronted with various conflicting expectations. You helped me to keep focused on the essential aspects of the work and encouraged me to pursue my own visions. Also many thanks go to my second copromotor Arjen. I could always talk to you when I was struggling with some theoretical problem or was doubting about details of some inference algorithm.

I also owe my gratitude to Rinus for arousing my enthusiasm for academic research. I especially enjoyed our collaboration in the time before I started the work presented in this thesis. Not to forget, I also thank you for being the chair of my doctoral thesis committee. I also owe thanks to the other members of this committee, Hendrick Blockeel and Vítor Santos Costa.

Working together on the METIS project with a variety of people, with a wide range of interests and backgrounds, was a very rewarding experience. I especially thank Dave Watts (TNO-ESI) for making sure that the project went smoothly, Bas Huijbrechts (TNO-ESI) for investing much time and effort in transferring the results of my work to an industrial setting and Tom Regelink (Altran) for doing an excellent job with gluing everything together, which allowed us to demonstrate our results as working prototype. I am also grateful for the collaboration with the other members of the project: Pieter Koopman (Radboud University), Teun Hendriks (TNO-ESI), Piërre van de Laar (TNO-ESI), Pierre America (TNO-ESI), Peter Novak (TU Delft), Roeland Scheepens (TU Eindhoven), Jesper Hoeksema (VU), Jan Laarhuis (Thales), Andre Bonhof (Thales) and Marcel Beekveld (Thales).

I spent some time at the KU Leuven during my time as PhD student, which was a great and instructive time. I especially thank Luc de Raedt, Jesse Davis and Joris Renkens for the collaboration. I also met a lot of other nice people and had inspiring conversations in Leuven. I want to thank Angelika, Dimitar, Gitte, Jan, Nimar, Wannes and many others.

I also owe thanks to the people I worked together with on *iTasks*, before I started my PhD project: Rinus, Pieter, Peter, Bas and Jeroen. Although I changed my topic quite a lot, the lessons learned and experiences from this earlier work, certainly helped me in my PhD project as well.

Also many other colleagues I did not mention before, contributed in making my time working at the Radboud University enjoyable. I especially thank Manxia for sharing the office with me for the last years of my PhD time. David, Faranak, Freek, Giso, Jurriën, László, Maarten, Marcos, Markus, Martijn, Paul, Sander, Thomas and everyone else I possibly forgot, I thank you for the great time. I also want to thank Ingrid for her support with organisational issues.

I am also very grateful for the support of my friends and especially thank Janna, Chris and Anna for sharing a flat with me during the time I worked on my PhD. You reminded my about the other important things in life, next to work. Janna and Chris, also thank you for being my paranymphs.

Am Ende möchte ich mich ganz besonders bei meiner Familie für ihre Unterstützung bedanken: Bei meinen Eltern Rita und Mick, bei meinem Bruder und bei meinen Großeltern.

CURRICULUM VITAE

Steffen Michels

1985	Born in Kleve, Germany
1996–2005	Johanna Sebus Gymnasium, Kleve
2005–2009	Bachelor Computing Science Radboud University Nijmegen
2008–2010	Master Computing Science Radboud University Nijmegen
2011	Junior Researcher Model-Based System Developement Institute for Computing and Information Sciences Radboud University Nijmegen
2011–2016	PhD-candidate Model-Based System Developement Institute for Computing and Information Sciences Radboud University Nijmegen