Learning Possibilistic Logic Theories

Cosimo Damiano Persia

Thesis for the degree of Philosophiae Doctor (PhD)
University of Bergen, Norway
2023

UNIVERSITY OF BERGEN

Learning Possibilistic Logic Theories

Cosimo Damiano Persia

Thesis for the degree of Philosophiae Doctor (PhD)
at the University of Bergen

Date of defense: 15.03.2023

Year: 2023

Title: Learning Possibilistic Logic Theories

Name: Cosimo Damiano Persia

Print: Skipnes Kommunikasjon / University of Bergen

Scientific environment

The work of this thesis, both research and writing, was done at the machine learning group at the *Department of Informatics* at the University of Bergen.

Parts of the project received funding from *L. Meltzers Høyskolefond*.

Acknowledgements

First, I am grateful to my family for having always been ready to listen to and help me these years. Special thanks to Renee, who I could always rely on and who allowed me to be in the perfect mindset to carry on this work.

Additionally, I have been lucky in meeting amazing colleagues in the machine learning group. I acknowledge Philip for having been the best office buddy. Thanks to Erlend for his 'board game nights' energy, Pierre for his incredibly accurate cynical comments, Victor for the pocket teas and controversial topic conversations, Natasha for the tastiest white beer selections, Ricardo for his LAN party and the best board game ever, Troels for knocking at everybody's door and reminding that it is lunchtime.

Moreover, I am honoured to have shared moments with fantastic friends. Thanks to Eugenio for the 'mozzarella from Milan', and for the true and honest friendship, Isaac for the deep discussions in the sauna, Alessandro for having shared with me the 'Telemark hype', and Eino for the chilly outdoor activities. I am grateful to Enrico and Francesca for having temporarily opened an 'exclusive' Italian restaurant in Bergen for me. Also, thanks to Sebastian and Kayne for their 'guttastemning', and Cecilie for her kindness and hospitality.

Last but not least, I am thankful to my supervisor Ana Ozaki and co-supervisor Ricardo Guimarães for their feedback concerning the work in this thesis.

Abstract in English

We address the problem of learning interpretable machine learning models from uncertain and missing information. We first develop a novel deep learning architecture, named RIDDLE (**R**ule **I**n**D**uction with **D**eep **LE**arning), based on properties of possibility theory. With experimental results and comparison with FURIA, a state of the art method, RIDDLE is a promising rule induction algorithm for finding rules from data. We then formally investigate the learning task of identifying rules with confidence degree associated to them in the *exact learning* model. We formally define theoretical frameworks and show conditions that must hold to guarantee that a learning algorithm will identify the rules that hold in a domain. Finally, we develop an algorithm that learns rules with associated confidence values in the *exact learning* model. We also propose a technique to simulate queries in the *exact learning* model from data. Experiments show encouraging results to learn a set of rules that approximate rules encoded in data.

Sammendrag

Vi tar opp problemet med å lære tolkbare maskinlæringsmodeller fra usikker og manglende informasjon. Vi utvikler først en ny dyplæringsarkitektur, RIDDLE: **R**ule **I**n**D**uction with **D**eep **LE**arning (regelinduksjon med dyp læring), basert på egenskapene til mulighetsteori. Med eksperimentelle resultater og sammenligning med FURIA, en eksisterende moderne metode for regelinduksjon, er RIDDLE en lovende regelinduksjonsalgoritme for å finne regler fra data. Deretter undersøker vi læringsoppgaven formelt ved å identifisere regler med konfidensgrad knyttet til dem i *exact learning*-modellen. Vi definerer formelt teoretiske rammer og viser forhold som må holde for å garantere at en læringsalgoritme vil identifisere reglene som holder i et domene. Til slutt utvikler vi en algoritme som lærer regler med tilhørende konfidensverdier i *exact learning*-modellen. Vi foreslår også en teknikk for å simulere spørringer i *exact learning*-modellen fra data. Eksperimenter viser oppmuntrende resultater for å lære et sett med regler som tilnærmer reglene som er kodet i data.

List of Publications

The results on this thesis are based on the following publications:

1. RIDDLE**: Rule Induction with Deep Learning** [Persia and Guimarães, 2023]

   Chapter 3 is based on the results of this work.

2. **On the Learnability of Possibilistic Theories** [Persia and Ozaki, 2020]

   Chapter 4 is based on the results of this work.

3. **Extracting Rules from Neural Networks with Partial Interpretations** [Persia et al., 2022]

   Chapter 5 is based on the results of this work.

4. **Extracting Horn Theories from Neural Networks with Queries and Counterexamples** [Persia and Ozaki, 2022]

   Chapter 5 is based on the results of this work.

Contents

List of Acronyms

FO *First Order*

KB *Knowledge Base*

CNF *Conjunctive Normal Form*

EL *Exact Learning*

PAC *Probably Approximately Correct*

AI *Artificial Intelligence*

ML *Machine Learning*

ANN *Artificial Neural Network*

MSE *Mean Squared Error*

SGD *Stochastic Gradient Descend*

Chapter 1

Introduction and Motivation

Machine learning is the branch of of *Artificial Intelligence* (AI) that studies the ability of acquiring knowledge from experience. The goal is to build systems capable of performing tasks after they have found patterns and induced knowledge from data. With the increase of the complexity of the tasks we would like AI systems to perform, follows an increase of the expressiveness of the knowledge induced from data. We want such knowledge to be interpretable. Works in the field of *computational logic* studied ways of representing knowledge, and performing reasoning with formal guaranteed properties. As learning, representing knowledge and reasoning are important abilities that interact with each other to perform complex tasks, combining the strengths of the approaches in these two fields complements some of their respective limitations.

Often, learning systems should have the ability to cope with conditions of ignorance to succeed in a given function. Consider the task of deciding what political party to vote for the next elections. Among many efforts, it requires acquiring knowledge (potentially contradictory) about different parties, reasoning about which one is expected to perform better, and drawing an (uncertain) conclusion about which party to support. As intelligent life developed mechanisms of dealing with lack of complete information, we would like to devise systems with similar skills. Making choices, planning, generalising knowledge from examples, etc. These abilities require some handling of uncertain information. Especially when exposed to ambiguous or contradictory facts. Therefore, practical AI systems should be able to learn and reason in situations of ignorance.

Moreover, complex AI systems should provide ways of expressing the reasons behind their own decisions. We have witnessed machine learning models winning against professional players of strategic-intensive games such as Chess and Go [Campbell et al., 2002; Hölldobler et al., 2017] and achieving other important milestones in pattern recog-

nition [Le et al., 2012], in natural language processing (e.g., when Deep Blue performed better than the champions of a quiz general knowledge game [Ferrucci, 2012]), and other areas. For some applications, like adding filters to a human face [Antipov et al., 2017], it may not be so relevant to fully understand why a model produced a particular output. However, when machine learning models are used to support high-stakes decisions, it is imperative to understand how predictions are made. For instance, whether loans should be approved [Sheikh et al., 2020], or whether defendants should be detained depending on the risk of offense [Kirkpatrick, 2017]. In particular, if rules are being derived from data, AI systems need to ensure that such rules are fair and can be trusted.

In the work of this thesis, we develop algorithms that learn patterns under the form of rules from imprecise data. We first describe a novel *Artificial Neural Network* (ANN) architecture, dubbed RIDDLE (**R**ule **I**n**D**uction with **D**eep **LE**arning). This architecture scales well with the amount of data and it provides an explainable model, as learned parameters can be converted to interpretable rules. Empirically, RIDDLE models show competitive performance, but no general guarantee on the learning outcome is available. For this reason, we formally study the learning task in the light of Angluin's *Exact Learning* (EL) model [Angluin, 1988]. We identify conditions that allow the learning task to always succeed, that is identify the target, an unknown concept. The proven positive results, allowed the development of Π_HORN*, an algorithm that finds (Horn) rules, scales well with the size and volume of the data, and is guaranteed to always identify the target and unknown rules.

In the following, we formalise the goal of this work (Section 1.1), then we clarify the problem setting and our contribution (Sections 1.2 and 1.3). In Section 1.4, we present the structure of this work.

1.1 Challenge: Learning Interpretable Models

The desire of understanding is closely related to learning processes. Consider the task of analysing data about complex chemical processes, weather prediction, social network analysis, etc. In similar cases, we are not only interested in knowing what will happen given a state of the environment, but we would like to discover the rules that hold in a specific domain, and give predictions based on interpretable models. These models should be expressive enough to represent complex concepts, and represent knowledge formally to guarantee explainability and unambiguous interpretations by different parties.

Knowing the reasons why things happen on a given domain can help learn more about the

problem itself. As we live in conditions of not perfect information, knowledge extracted from data can help in reasoning why a generated model fails. For instance, if a system that scores users depending on their ability to pay back a loan provides explanation about the predicted score, it can be used to check if the prediction is to be trusted. In case an error in the model is identified, this can reveal a problem in the data, in societal structure, etc.

The benefit of a descriptive model is not only related to the phenomena it describes, but such models can contribute to solutions of other problems. Knowledge transfer is one activity that improves companies expertise, and it is only possible through the availability of knowledge artifacts like documentation, or interpretable rules. The understanding of a given problem positively contributes to the discovery of solutions of more complex problems through the common effort of many persons. This is especially relevant for solving global challenges.

Systems that find patterns from data should have properties that allow a constructive influence on human activities. Some of them are:

- **Fairness**: Ensuring that predictions are not based on statistical correlations in the available data. Learned models should not be biased against underrepresented groups. An interpretable model can express the reasons why a person has been denied the opportunity to get a loan, and it is possible to correct decisions based on unwanted correlations.
- **Robustness**: Ensuring that noisy and small perturbations of the input to the model do not lead to large changes in the prediction. This problem is common in algorithms that identify undesired numerical patterns in data. Often, it leads to security problems. Indeed, such venerability can be used by an attacker to take advantage of the system by obtaining desired (and false) predictions.
- **Causality:** Ensuring that the generated model contains only patterns describing causal relationships between concepts in the data.

Many works [Ozaki, 2020a] tackled the problem of learning a formal representation about the relationships between concepts in a specific domain using classical logic-based knowledge representation formalisms. In this way, the patterns found in data are expressed with an unambiguous and clear language that both humans and machines can interpret. A major difficulty is that in real world applications classical logic is too rigid to cope with conditions of ignorance and manage imprecise and incomplete information.

Indeed, solving learning tasks requires the ability to cope with many facets of uncertainty.

Consider the task of selecting the best generated model suited for the data at disposal, learning from a dataset populated by many imprecise measurements of unreliable sensors, or learning from a dataset that expresses relationships between objects and attributes that omits some assertions (for example 'NULL' values in database tables). For this reason, we need to adopt a knowledge representation formalism that is able to manage and express uncertain information.

1.2 Problem Setting

We formulate the problem of learning interpretable rules expressed in possibilistic logic [Dubois et al., 1994] from data. An expressive knowledge representation language designed to represent uncertainty and manage inconsistent knowledge.

Possibilistic Logic

There are different approaches to dealing with uncertainty [Parsons and Hunter, 1998]. *Possibilistic logic* [Dubois et al., 1994] is a promising formalism for approaching it. In this language, we can express relationships between concepts and express subjective belief on the truth value associated to such statements. We express formulas in possibilistic logic as a pair (ϕ, α) where ϕ asserts a statement and α denotes the belief on ϕ. The value of α ranges in the interval $[0, 1]$ and the closer it is to 1, the stronger the belief becomes. For instance, a student in biology can be uncertain whether a whale is a mammal or an amphibian with a preference of assigning whales to be mammals. We can express this knowledge in possibilistic logic with the three constraints:

$$(\mathsf{whale} \rightarrow \mathsf{mammal}, 0.9) \tag{1.1}$$

$$(\mathsf{whale} \rightarrow \mathsf{amphibian}, 0.2) \tag{1.2}$$

$$(\mathsf{mammal} \leftrightarrow \neg\mathsf{amphibian}, 1) \tag{1.3}$$

Formulas 1.1 and 1.2 denote that the student is more confident that a whale is a mammal as it assigns a low belief that a whale is an amphibian. Formula 1.3 expresses the implicit and sure knowledge that mammals cannot be amphibian and vice-versa. This formalism gives freedom to show a graded notion of uncertainty and makes a clear distinction between the concepts of truth and belief [Dubois and Prade, 2001]. A statement can be either true or false like in classical logic and the belief degree acts on the meta-level.

Exact Learning

To formally study the learning task, we employ definitions of computational learning theory [Kearns and Vazirani, 1994] that studies in mathematical terms the process of acquiring knowledge. We mainly investigate properties of algorithms devised for learning possibilistic logic formulas with the EL model [Angluin, 1988] where there is a learner that forms concepts by asking queries. In this model, it is assumed that:

- there is a *teacher* that knows the target and unknown concept (in this work a set of possibilistic logic formulas). Moreover, the teacher can answer questions concerning the structure of the target;
- there is a *learner* that can ask queries to a teacher to acquire knowledge and identify the target concept.
- The types of queries the learner can ask is fixed and well-defined. Also, the teacher and the learner share the information concerning the vocabulary and expressiveness of the target.

This model is general and applicable to diverse specific instances. We illustrate this statement with Example 1.1, where we describe the learning task of a chef that learns how to make cream with a new machine.

Example 1.1. Consider a pastry chef who is using a professional machine to prepare whipped cream but the outcome of the process is too liquid. The chef can consider some causes for this to happen: not enough time or too much time was spent to whip the cream, or the ingredients were expired. In the EL model, the chef would play the role of the learner who will try to identify the reason why the whipped cream is too liquid by using the machine. The actions of the chef can be considered as queries to the machine, that plays the role of the teacher, and that will provide the desired cream if the chef executes the right steps. The quality of the outcome can represent the answer to such query/action. In this setting, the goal of the chef is to identify the right step to perform to obtain the desired whipping cream based on the outcome returned by the machine.

Moreover, if possibilistic logic is used to represent the acquired knowledge, one can model cases of complete uncertainty. Both low-whipping and high-whipping time, which are two mutually exclusive conditions, can be considered fully possible. On the other hand, if the uncertainty in this scenario is modelled by probability theory the more the chef believes that low-whipping time is the cause for the cream to be liquid, the less it considers the case of high-whipping time.

The flexibility of the uncertainty measure used in possibilistic logic can capture other facets of uncertainty. Assume that the chef knows the quality of the ingredients and the condition that the ingredients are expired is considered to be less possible, e.g. associated with the value 1/3. In probability theory, complete ignorance of the first two conditions would make us assign probability 1/3 to every condition (Laplace criterion). Thus, it would not model the knowledge about the quality of the ingredients and the ignorance about the time spent to whip the cream. ◁

The most studied communication protocol in this model allows the learner to ask questions of two kinds, called *membership* and *equivalence* queries. Membership queries allow the learner to know whether a certain statement holds (e.g. 'Is the timer of the whipped cream maker broken?') and its answer can be 'yes' or 'no'. Equivalence queries allow the learner to check whether a hypothesis (e.g. an explanation of why the whipped cream maker did not work as expected) is correct and, if not, to fix it using a counterexample. In Example 1.1, the chef wants to learn how to use the machine, and the latter may play the role of the teacher. The answer to the membership query 'Is the timer of the whipped cream maker broken?' can be found when the chef empirically tests the timer with another trusted clock. Equivalence queries happen when the chef formulates a hypothesis to explain the cause of the problem and takes specific actions on the machine to either confirm the hypothesis or find a counterexample. If the outcome of the process does not correspond to the desired one, it may represent a counterexample. Other types of queries, such as *superset* and *subset* queries, are also considered in this work.

1.3 Contribution

We propose a novel ANN architecture dubbed RIDDLE that finds patterns expressed as possibilistic logic rules from data. Since it is based on the formal theory of possibilistic logic, this architecture generates models that are both able to handle uncertain information during the training phase, and provide certainty about the given prediction. As RIDDLE models allow to check if predictions can be trusted, their adoption can give benefits in those scenarios where decisions must be taken based on these predictions. We also formally describe the semantics of the parameters of the network. As a consequence, we show an algorithm that decodes the optimised parameters after the training phase into interpretable possibilistic rules. This feature allows RIDDLE models to be used in scenarios where predictions are used to give suggestions in high-stakes decision making tasks. We carry empirical experimental evaluations on 15 datasets freely available at the UCI machine learning repository [Dua and Graff, 2017]. With a comparison between

trained RIDDLE models with state of the art fuzzy rule induction models, we can say that RIDDLE is a competitive method with potential to be used in real life scenarios.

The performance of RIDDLE is promising, but to formally study the learning task of identifying possibilistic logic rules, we base our work on notions of computational learning theory [Kearns and Vazirani, 1994]. We first define the concept of a *learning system*: a formal framework that takes into account notions of the theory of computation [Sipser, 1997] and computational learning theory. We identify conditions that must hold to guarantee that an EL learner is able to identify target rules by asking a predefined set of queries. We also identify negative results, and in both cases we formally prove our statements. Additionally, we show the conditions that must hold for using algorithms developed for finding classical rules, into algorithms that learn possibilistic logic rules. In this way, the abundant number of *classical* learning algorithms developed, and present in the literature, becomes relevant in scenarios with some degrees of uncertainty.

Finally, based on the positive learnability results we proved, we developed Π_HORN*, an EL algorithm that is guaranteed to identify possibilistic Horn[1] logic rules. This algorithm runs in polynomial time with respect to the symbols in the considered language and the number of rules $\mathcal{K}$ holding in a given domain, and it outputs a set of possibilistic logic rules equivalent to $\mathcal{K}$. Π_HORN* is an adaptation of the LRN algorithm developed by Frazier and Pitt [1993a] that learns classical Horn rules. To test the efficacy of the new methodology, we propose a new approach for extracting rules hidden in so-called black-box machine learning models, where neural network models lie in. First, we convert a given dataset into a binarized form and train a neural network (there are no assumptions regarding the internal architecture of the neural network). We then employ the Π_HORN* algorithm for learning possibilistic Horn rules of the form $((\mathsf{sunny} \wedge \mathsf{warm}) \rightarrow \mathsf{hike}, 0.8)$. As Π_HORN* is developed in the EL model, it will act as a learner that asks queries to a teacher in order to learn an abstract target (in our case a set of rules). We treat a trained neural network model as the teacher, as originally proposed by Weiss et al. [2018] for extracting automata.

However, answering efficiently queries in the EL model is not always possible. In our case, simulating equivalence queries asked by Π_HORN* with an ANN is hard. We propose an efficient technique that can provide probabilistic guarantees for the correctness of the rules extracted. This method is based on the connection between the EL and the *Probably Approximately Correct* (PAC) learning models [Angluin, 1988; Valiant, 1984].

[1] A syntatic restriction of full propositional logic.

1.4 Structure of the Thesis

This work is divided in seven chapters:

- Chapter 2 – Preliminaries: we introduce the syntax and semantics of classical propositional logic while providing relevant definitions applicable for the more expressive *First Order* (FO) logic. Then, we describe the main properties of possibility theory, and the type of uncertainty it captures. Based on these definitions, we introduce possibilistic logic. We discuss syntax, and semantics, and its properties in expressing uncertainty and handling inconsistent knowledge. Moreover, we state reasoning complexity results and key properties that are going to be used in later chapters. Then, we provide notions of computational learning theory, and important definitions of the EL and PAC model such as: learning framework, learner, teacher, query, and learnability. Finally, we introduce basic notions of ANN models, including general architecture and training algorithm.

- Chapter 3 – RIDDLE: Rule Induction with Deep Learning: We prove properties based on possibility theory that provide ways of propagating uncertainty. With these results, we define the RIDDLE architecture whose parameters can be optimised with standard gradient based algorithms. We explain how weights of a trained RIDDLE model can be translated into interpretable (possibilistic) rules. Moreover, we provide results of the empirical evaluation of RIDDLE with 15 different dataset available at the UCI machine learning repository [Dua and Graff, 2017]. We also compare the results with a state of the art fuzzy rule induction algorithm. We claim that RIDDLE is a competitive model. We conclude with a summary of the content of the chapter, and future works that may improve the applicability of RIDDLE models. This chapter is based on the results of the work published at NLDL23 [Persia and Guimarães, 2023].

- Chapter 4 – Exact Learning of Possibilistic Logic Theories: we study whether possibilistic theories are learnable in the EL model. We first give a general and formal definition of a learning context, denoted *learning system*, that takes into account notions of the theory of computation. With this formal framework, we prove under which conditions it is possible to guarantee exact identification of an unknown target concept in the EL model. We show under which circumstances it is possible to reduce polynomial time learnability results of classical logic formulas to polynomial time learnability of possibilistic logic formulas and vice-versa. At the end of the chapter, we mention proven results that hold in the PAC model due to the connection between the EL and the PAC model. We conclude with a

discussion of proven results, and open challenges. This chapter is based on the results of the work published at IJCAI20 [Persia and Ozaki, 2020].

- Chapter 5 – Exact Learning Possibilistic Horn Theories: we first develop HORN*, an EL learning algorithm that identifies unknown target Horn formulas by asking queries. We propose a general method that answers queries asked by EL algorithms based on the information available from data. We test this approach by treating an ANN model trained on data as a teacher that answer queries. Then, using the positive learnability results in the previous chapter, we develop Π_HORN*, a learning algorithms that identifies unknown target possibilistic Horn formulas by asking queries as defined in the *Exact Learning* (EL) model. Also in this case, we test our method by extracting possibilistic Horn rules from trained ANN models from uncertain data. We conclude the chapter with a discussion about the generality of our work that can inspire related research. This chapter is based on the results published at NLDL22 [Persia et al., 2022] and KR4HI [Persia and Ozaki, 2022].

- Chapter 6 – Related Work: it is divided in three parts. In the first, we mention relevant work concerning rule induction algorithms and neurosymbolic AI. In the second part, we discuss works in the EL field that focused on algorithms that exactly identify logic formulas. In the third part, we present uncertainty measures alternative to possibility theory and similar knowledge representation formalisms like possibilistic logic.

- Chapter 7 – Conclusion: we discuss the contribution and main results developed during this work. At the end, we introduce ideas that can complement or extend the methods, and solutions proposed in this work.

- Finally, in Appendix A, the reader can find omitted proofs and definitions.

Chapter 2

Preliminaries

We present basic notions that are going to be used for the rest of this work. In Section 2.1, we introduce notions of propositional logic and mention the required notation of *First Order* (FO) logic for a comprehensive understanding of this work. A more detailed presentation of FO logic can be found in Appendix A.1. In Section 2.2, we introduce possibility theory, that is the theoretical foundations of possibilistic logic. We present the latter in Section 2.3 together with some key properties that are useful to formally explain and show results in all next chapters. In Section 2.4, we define notions of computational learning theory used to define the learning settings formally and prove learnability and transferability results. These notions are relevant for Chapters 4 and 5. Finally, in Section 2.5, we present artificial neural networks; a prominent algorithm in machine learning for finding patterns in data. Such algorithm is relevant for Chapters 3 and 5.

2.1 Knowledge Bases and Satisfiability

Knowledge can be expressed in many forms. *Propositional logic* is a formal language primarily used to mathematically describe statements using propositions. Moreover, this language is more easily processable by algorithms, compared to natural language statements. We formally define it here as we use it throughout this entire work for explaining proofs, present algorithms, and clarify definitions with examples.

An alphabet Σ of propositional logic language consists of

- a finite set $\mathsf{V} \coloneqq \{\mathsf{v}_1, \mathsf{v}_2, \cdots\}$ of propositional variables;
- the set of connectives $C \coloneqq \{\neg/1, \wedge/2, \vee/2\}$; and

- the special characters "(", ")".

A *propositional atomic formula* is a variable in V. Each variable in V is associated a "proposition" that can be either true or false. By conjoining them with connectives, we can express increasingly complex statements. The set of *propositional formulas* $\mathfrak{L}_{\mathsf{V}}$ satisfy the following properties:

- the special symbols "true", and "false" are always in the language, $\top, \bot \in \mathfrak{L}_{\mathsf{V}}$;
- every propositional atomic formula ϕ is in $\mathfrak{L}_{\mathsf{V}}$;
- If $\phi \in \mathfrak{L}_{\mathsf{V}}$, then $\neg\phi \in \mathfrak{L}_{\mathsf{V}}$;
- If $\circ/2 \in \{\wedge, \vee\}$, then $\phi, \psi \in \mathfrak{L}$ implies $(\phi \circ \psi) \in \mathfrak{L}_{\mathsf{V}}$.

We omit the subscript in $\mathfrak{L}_{\mathsf{V}}$ if it is clear from the context. We additionally give names to formulas obeying specific constraints. A *literal* over V, denoted with the symbol l, is either a *variable* $\mathsf{v} \in \mathsf{V}$ or its negation, in symbols, $\neg\mathsf{v}$. The former is also called a *positive literal*, the latter a *negative literal*. *A clause* over V is a disjunction ($\vee$) of literals over V. It is called *Horn* if at most one literal is positive. *A* Conjunctive Normal Form *(CNF) formula* over V is a conjunction $\wedge$ of clauses over V. A formula is *Horn* CNF if all its clauses are Horn. We may omit 'over V' in formulas, clauses, and literals. A clause ψ can also be expressed as a *rule* r of the form $\mathsf{ant}(\mathsf{r}) \rightarrow \mathsf{con}(\mathsf{r})$ where $\mathsf{ant}(\mathsf{r})$ (the *antecedent* of r) is the set of all negated literals in ψ and $\mathsf{con}(\mathsf{r})$ (the *consequent* of r) is the single positive literal in ϕ, or empty.

Example 2.1. *Let* V *be the set* $\{\mathsf{v_w}, \mathsf{v_j}, \mathsf{v_d}, \mathsf{v_h}\}$ *where variables are assigned the propositions 'It is a working day', 'the job is unfinished', 'there is an imminent deadline', and 'it is a holiday', respectively. The clause* $(\mathsf{v_h} \vee \mathsf{v_w})$ *states that at least one the propositions appearing in it must be true, that is 'either is holiday or it is a working day'. This clause can be conjoined with* $(\neg\mathsf{v_d} \vee \neg\mathsf{v_j} \vee \mathsf{v_w})$ *to form the CNF formula* $((\mathsf{v_h} \vee \mathsf{v_w}) \wedge (\neg\mathsf{v_d} \vee \neg\mathsf{v_j} \vee \mathsf{v_w}))$. *Moreover, the clause* $(\neg\mathsf{v_d} \vee \neg\mathsf{v_j} \vee \mathsf{v_w})$ *can be expressed as a rule of the form* $\mathsf{r} \coloneqq (\mathsf{v_d} \wedge \mathsf{v_j} \rightarrow \mathsf{v_w})$ *where* $\mathsf{ant}(\mathsf{r}) = \{\mathsf{v_d}, \mathsf{v_j}\}$ *and* $\mathsf{con}(\mathsf{r}) = \{\mathsf{v_w}\}$. *Intuitively, it means 'if there is an imminent deadline and the job is unfinished, then it is a working day'. As such clause is Horn, there is only one way of expressing it as a rule with the consequent as a positive literal.* ◁

For a conjunction of clauses ϕ, *the set of subformulas* of ϕ is the smallest set of formulas $\mathcal{S}_\phi$ satisfying the following conditions:

- The formula ϕ is a subformula of itself, that is, $\phi \in \mathcal{S}_\phi$;

- If $\neg\psi \in \mathcal{S}_\phi$, then $\psi \in \mathcal{S}_\phi$;
- If for any connective $\circ/2 \in C$, we have $(\psi_1 \circ \psi_2) \in \mathcal{S}_\phi$, then $\psi_1, \psi_2 \in \mathcal{S}_\phi$

A (finite) set of statements, or alternatively a *Knowledge Base* (KB), gives a description of the constraints that hold in a given domain using propositions in V. We denote KBs with calligraphic letters, such as $\mathcal{H}$, $\mathcal{K}$, or $\mathcal{T}$, with superscript if more than one is involved in our discussion. We treat a KB $\mathcal{H}$ as a conjunction of formulas $\mathcal{H} \coloneqq \bigwedge_{i=0}^{n} \phi_i$ with $\phi_i \in \mathfrak{L}$. By definition of $\mathfrak{L}$, it follows that $\mathcal{H} \in \mathfrak{L}$. For convenience, we will interchangeably denote $\mathcal{H}$ as a set of formulas $\{\phi_i \in \mathfrak{L} \mid 0 \leq i \leq n\}$. We recall that a conjunction of formulas can be transformed into a logical equivalent CNF formula [Bell and Machover, 1977]. The size of a formula $\phi \in \mathfrak{L}$, denoted $|\phi|$, is the number of symbols in the language used to express ϕ. Similarly, the size of a $\mathcal{H}$ is $|\mathcal{H}| = |\bigwedge_{i=0}^{n} \phi_i|$.

Example 2.2. *The subformulas of* $\phi \coloneqq ((\neg\mathsf{v}_2 \vee \mathsf{v}_1) \wedge (\neg\mathsf{v}_1 \wedge \mathsf{v}_3))$ *are: itself,* $(\neg\mathsf{v}_2 \vee \mathsf{v}_1)$, $(\neg\mathsf{v}_1 \wedge \mathsf{v}_3)$, $\neg\mathsf{v}_2$, v_2, v_1, $\neg\mathsf{v}_1$, *and* v_3. *If in our discussion* ϕ *is a KB, we may represent it as the set* $\{(\neg\mathsf{v}_2 \vee \mathsf{v}_1), (\neg\mathsf{v}_1 \wedge \mathsf{v}_3)\}$. *The size of the formula* $(\neg\mathsf{v}_2 \vee \mathsf{v}_1)$ *is* 6 *because the symbols in this formulas are* $\mathsf{v}_1, \mathsf{v}_2, \neg, \vee, \text{`('}, \text{`)'}$. ◁

So far, we have shown a formal syntax to our language $\mathfrak{L}$. We now define a formal semantics for a propositional logic language through the notion of truth-value assignment. An *interpretation* over V is a function $\mathcal{I} : \mathsf{V} \to \{\top, \bot\}$ that maps every variable either to the 'true' ($\top$) or 'false' ($\bot$) truth value. An interpretation $\mathcal{I}$ for a language $\mathfrak{L}$ *satisfies* a formula $\phi \in \mathfrak{L}$ iff $\mathcal{I}$ does not violate the constraints imposed by ϕ. If an interpretation $\mathcal{I}$ satisfies ϕ, we write $\mathcal{I} \models \phi$, otherwise $\mathcal{I} \not\models \phi$. We say that $\phi \in \mathfrak{L}$ is *satisfiable* if there is an interpretation $\mathcal{I}$ that satisfies ϕ. Moreover, ϕ is *falsifiable* if its negation $\neg\phi$ is satisfiable. A *tautology* or a *valid formula* is a formula that is not falsifiable.

Example 2.3. *Let* $\phi \coloneqq (\neg\mathsf{v}_\mathsf{d} \vee \neg\mathsf{v}_\mathsf{j} \vee \mathsf{v}_\mathsf{w})$ *be the clause presented in Example 2.1. Any interpretation that maps the value of any literal in* ϕ *to* $\top$ *satisfies it. Any interpretation* $\mathcal{I}$ *such that* $\mathcal{I}(\mathsf{v}_\mathsf{d}) = \top$, $\mathcal{I}(\mathsf{v}_\mathsf{j}) = \top$, *and* $\mathcal{I}(\mathsf{v}_\mathsf{w}) = \bot$ *falsifies* ϕ. ◁

It is not always reasonable to assume complete information in learning settings. Therefore, we generalised the notion of interpretation that allows for variables to have an unknown truth value. A *partial interpretation* $\mathcal{I}^*$ over V is a function $\mathcal{I}^* : \mathsf{V} \to \{\top, \bot, ?\}$ that states which variables are regarded as 'true', 'false', and 'unknown'. $\mathcal{I}^*$ *falsifies* a variable $\mathsf{v} \in \mathsf{V}$ if $\mathcal{I}^*(\mathsf{v}) = \bot$, otherwise $\mathcal{I}^*$ *satisfies* it. $\mathcal{I}^*$ satisfies a negative literal $\neg\mathsf{v}$ iff $\mathcal{I}^*(\mathsf{v})$ is equal to '?' or '$\bot$'. $\mathcal{I}^*$ satisfies a clause ϕ if it satisfies at least one literal in ϕ. In other words, $\mathcal{I}^*$ satisfies a clause if there is a way of replacing 'unknown' values (?) with 'true' ($\top$) or 'false' ($\bot$) such that the clause is satisfied. We denote in symbols

$\mathcal{I}^* \models \phi$, if $\mathcal{I}^*$ satisfies ϕ. We write $\mathcal{I}^* \not\models \phi$ instead, if $\mathcal{I}^*$ does not satisfies ϕ. With this definition, interpretations can be considered as a special case of partial interpretations where no variable is mapped to the unknown value.

A KB $\mathcal{H}$ is satisfiable if there is an interpretation that satisfies every formula in $\mathcal{H}$ and it is falsifiable if there is an interpretation that does not satisfy at least a formula in $\mathcal{H}$. If there is not interpretation that satisfies $\mathcal{H}$, then we say that $\mathcal{H}$ is *inconsistent*. If every interpretation that satisfies $\mathcal{H}$ satisfies also a formula $\phi \in \mathfrak{L}$, then we say that $\mathcal{H}$ entails ϕ, written $\mathcal{H} \models \phi$, otherwise we write $\mathcal{H} \not\models \phi$. If $\mathcal{H}$ entails every formula in another KB $\mathcal{H}'$, we write $\mathcal{H} \models \mathcal{H}'$. Two KBs $\mathcal{H}, \mathcal{H}'$ are *logical equivalent* if $\mathcal{H} \models \mathcal{H}'$ and $\mathcal{H}' \models \mathcal{H}$, also denoted with $\mathcal{H} \equiv \mathcal{H}'$. A KB is *non-trivial* if it is satisfiable and falsifiable.

Example 2.4. *Let* V *be the set* $\{\mathsf{v_w}, \mathsf{v_j}, \mathsf{v_d}, \mathsf{v_h}\}$ *and consider the CNF formula* $\phi := ((\mathsf{v_h} \vee \mathsf{v_w}) \wedge (\neg\mathsf{v_d} \vee \neg\mathsf{v_j} \vee \mathsf{v_w}))$ *given in Example 2.1. The partial interpretation* $\mathcal{I}_1 = \{(\mathsf{v_w}, \bot), (\mathsf{v_j}, \top), (\mathsf{v_d}, ?), (\mathsf{v_h}, ?)\}$ *satisfies* ϕ *because we can replace the assignment* $\mathcal{I}_1(\mathsf{v_h}) = ?$ *to* $\mathcal{I}_1(\mathsf{v_h}) = \top$ *and* $\mathcal{I}_1(\mathsf{v_d}) = ?$ *to* $\mathcal{I}_1(\mathsf{v_d}) = \bot$ *such that every clause in* ϕ *is satisfied by* $\mathcal{I}_1$*. On the contrary* $\mathcal{I}_2 = \{(\mathsf{v_w}, \bot), (\mathsf{v_j}, ?), (\mathsf{v_d}, ?), (\mathsf{v_h}, \bot)\}$ *does not satisfy* ϕ *as the first clause is always falsified; it does not matter to which values* $\mathsf{v_j}, \mathsf{v_d}$ *are assigned to. The KB* $\{\neg\mathsf{v_w}, (\neg\mathsf{v_d} \vee \mathsf{v_w}), (\mathsf{v_d} \vee \mathsf{v_w})\}$ *is inconsistent because there is no interpretation that assigns to the variable* $\neg\mathsf{v_d}$ *both the value* $\top$ *and* $\bot$*.* ◁

In Chapter 4, we are going to discuss theoretical results that include also the more expressive *First Order* (FO) logic language (also denoted by $\mathfrak{L}$ depending on the context) consisting of closed well-formed formulas in the classical sense. We refer to Appendix A.1 for a quick definition of FO logic. We point to the book written by Bell and Machover [1977] for a comprehensive formalisation. Also in FO logic, we denote by $\neg, \wedge, \vee$ the negation, conjunction, and disjunction operators, and by $\forall, \exists$ the universal and existential quantifiers as usual. When we introduce an FO language we mean any fragment of FO logic, also propositional logic. We denote by $\Sigma_{\mathcal{H}}$ the vocabulary of the KB $\mathcal{H}$, i.e. the set of predicates (or propositional variables) occurring in $\mathcal{H}$. We keep the same notation for interpretations of an FO language and use the same notation for satisfiability of an FO formula with respect to an FO partial interpretation.

2.2 Possibility Theory

The handling of uncertainty has been an issue in many scientific disciplines. Machine learning is one of them, because often, learning tasks involve information that is uncertain and incomplete. The demand to handle uncertainty led to the development of many

formalisms like probability theory [Jaynes, 2003], imprecise probabilities [Walley, 1991], and evidence theory [Shafer, 1976]. Among the uncertainty handling formalisms in the literature, possibility theory [Dubois and Prade, 2014] stands out for its simplicity and its freedom in expressing any case of lack of information.

Let Ω be the set of states of affair. For instance, the set of interpretations over a logic language $\mathfrak{L}$. A possibility distribution $\pi : \Omega \to [0, 1]$ is a function that maps an element in Ω to a possibility value. For an $\mathcal{I} \in \Omega$, the closer $\pi(\mathcal{I})$ is to 0, the less possible the event described by $\mathcal{I}$ is considered. On the other hand, $\pi(\mathcal{I}) = 1$ denotes that $\mathcal{I}$ is fully possible. Let the set of states $A \subseteq \Omega$ describe an assertion or a proposition (like the set of models of a formula). We define the *possibility* $\Pi_\pi(A)$ measure that indicates how much A is coherent with π and the *necessity* measure $N_\pi(A)$ that expresses the *certainty degree of* A being implied by π as follows:

$$\Pi_\pi(A) \coloneqq \sup_{\mathcal{I} \in A}\{\pi(\mathcal{I})\}; \quad \text{and} \quad N_\pi(A) \coloneqq \inf_{\mathcal{I} \in \Omega \setminus A}(1 - \pi(\mathcal{I})) = 1 - \Pi(A^c).$$

For sets $A, B \subseteq \Omega$, the possibility measure satisfies the *maxitivity* property and the necessity measure its dual, *minitivity* [Dubois et al., 1994]:

$$\Pi_\pi(A \cup B) = \max(\Pi_\pi(A), \Pi_\pi(B)); \qquad \textbf{(Maxitivity)}$$
$$N_\pi(A \cap B) = \min(N_\pi(A), N_\pi(B)). \qquad \textbf{(Minitivity)}$$

If clear from the context, we omit the subscript π. Being certain of both A and A^c in possibility theory implies that neither A nor A^c are fully possible according to a possibility distribution π, expressing a type of inconsistency. Additionally, it always holds that $\Pi(A \cap B) \leq \min(\Pi(A), \Pi(B))$ meaning that adding constraints to a system may decrease the possibility of that system to be compliant with future observations. The more specific we are in describing a state, the less possible it might become. Moreover, we also have the inequality $N(A \cup B) \geq \max(N(A), N(B))$. Intuitively, we can be certain about a state in which conditions A or B hold, but we are less certain if we consider these conditions separately. In possibility theory we have the freedom of neither believing in a set of statements A nor its negation A^c, while this is not allowed by the duality of a probability measure (that is $\mathcal{D}(A) = 1 - \mathcal{D}(A^c)$).

We recall a connection between probability and the possibility/necessity measures as the latter can be used to model imprecise probabilities [Dubois and Prade, 1992]. Let $\mathcal{D}$ be a probability distribution $\mathcal{D} : \Omega \to [0, 1]$ such that $\Sigma_{A \in \Omega}\mathcal{D}(A) = 1$. A possibility distribution π induces a family of probability measures P such that every probability

measure $\mathcal{D} \in \mathsf{P}$ over $A \in \Omega$ satisfies the constraint:

$$N(A) \leq \mathcal{D}(A) \leq \Pi(A) = 1 - N(A^c).$$

A wide interval associated to A implies a high uncertainty on the states in A. Figure 2.1 depicts the relationship between the possibility and necessity measure.

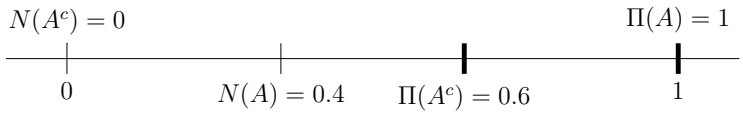

Figure 2.1: Graphical representation of the dependency of necessity and possibility of an element $A \in \Omega$ and its negation when $\Pi(A) = 1$ and $\Pi(A^c) = 0.6$.

We refer to Appendix A.4 for more detail about this connection.

2.3 Possibilistic Logic

Possibility theory has a logical counterpart named *possibilistic logic* [Dubois and Prade, 2014]. It remains close to classical FO logics while being able to express uncertainty over truth values. Possibilistic logic belongs to the family of weighted logics and it associates a certainty degree to classical formulas. Let $\mathfrak{L}$ be an FO logic language. A *possibilistic formula* is a pair (ϕ, α) where $\phi \in \mathfrak{L}$ (restricted to well-formed formulas without free variables) and α is a value in the interval $(0, 1]$ with finite precision, also called the *valuation* of ϕ. A *possibilistic KB* $\mathcal{H}$ is a finite set of possibilistic formulas. Intuitively, if $(\phi, \alpha) \in \mathcal{H}$, this means that ϕ is believed to be true at least to degree α. The closer α gets to 1, the more ϕ is believed to be certain. A possibilistic KB $\mathcal{H}$ can be seen as a stratified collection of classical formulas where each "layer" is associated a different valuation. Each layer or level of this stratification corresponds to a set of formulas that are considered certain with the same degree. The upper layers correspond to the set of formulas considered more certain. With the operators α-cut and $\overline{\alpha}$-cut with $\alpha \in (0, 1]$ we can ignore some layers of a KB $\mathcal{H}$ as follows:

$$\mathcal{H}_\alpha = \{(\phi, \beta) \in \mathcal{H} \mid \beta \geq \alpha\}, \quad \mathcal{H}_{\overline{\alpha}} = \{(\phi, \beta) \in \mathcal{H} \mid \beta > \alpha\}.$$

The *classical projection* of a possibilistic KB $\mathcal{H}$ is denoted by $\mathcal{H}^* = \{\phi \mid (\phi, \alpha) \in \mathcal{H}\}$ that is the corresponding KB expressible in $\mathfrak{L}$ by dropping all valuations of formulas in $\mathcal{H}$. It will be convenient to define the set $\mathcal{H}^v = \{\alpha \mid (\phi, \alpha) \in \mathcal{H}\}$ that contains all the valuations present in $\mathcal{H}$. For every valuation $\alpha \in (0, 1]$, by combining α-cut

and the classical projection, we can obtain the classical projection of some "layers" of a possibilistic KB with $\mathcal{H}^*_\alpha$.

The semantics of a possibilistic logic formula depends on the semantics defined for the underlying logic, that gives meaning to the classical formulas, plus the additional notion of uncertainty modelled by a possibilistic measure. The semantics of a classical KB partitions the set of interpretations in two sets: the set of interpretations that satisfy the KB and the set of interpretation that do not. A possibilistic KB does not partition the set of interpretations but it defines a possibility distribution over the interpretations expressing a preference relation over them. More precisely, let Ω be a countable (possibly infinite) set of interpretations of the language $\mathfrak{L}$. We know that a possibility distribution $\pi : \Omega \to [0, 1]$ associates to each interpretation in Ω a *possibility value* such that for $\mathcal{I} \in \Omega$, the closer $\pi(\mathcal{I})$ is to 1, the more the interpretation is considered possible according to π. For $\mathcal{I}_1, \mathcal{I}_2 \in \Omega$, having $\pi(\mathcal{I}_1) > \pi(\mathcal{I}_2)$ means that $\mathcal{I}_1$ is preferred over $\mathcal{I}_2$.

It is convenient, from a knowledge representation perspective, to express the possibility and necessity measures of formulas instead of interpretations. For this reason, we extend the definition of such measures. For any formula $\phi \in \mathfrak{L}$, we characterize the possibility measure Π_π and necessity measure N_π induced by the possibility distribution π:

$$\Pi_\pi(\phi) = \sup_{\mathcal{I} \in \Omega} \{\pi(\mathcal{I}) \mid \mathcal{I} \models \phi\} \quad \text{and} \quad N_\pi(\phi) = 1 - \Pi_\pi(\neg\phi) = \inf_{\mathcal{I} \in \Omega} \{1 - \pi(\mathcal{I}) \mid \mathcal{I} \not\models \phi\}.$$

This definition is a cleaner alternative notation for the measures (defined in Section 2.2) $\Pi_\pi([[\phi]])$ and $N_\pi([[\phi]])$ where $[[\phi]] \coloneqq \{\mathcal{I} \in \Omega \mid \mathcal{I} \models \phi\}$ is the set of all interpretations that satisfy ϕ with respect to the possibility distribution π. If clear from the context, we omit the subscript π from Π_π and N_π. $\Pi(\phi)$ outputs the highest valuation of an interpretation that satisfies ϕ. If ϕ is not satisfiable, then $\Pi(\phi) = 0$ and if ϕ is valid, then $\Pi(\phi) = \sup\{\pi(\mathcal{I}) \mid \mathcal{I} \in \Omega\}$ [Dubois et al., 1994]. The formula ϕ is more certainly true when $N(\phi)$ is closer to 1, that is when $\neg\phi$ is less possible according to the possibility distribution. A possibility distribution π is *normalised* if it assigns to at least an element $\mathcal{I} \in \Omega$ the maximal possibility value, $\pi(\mathcal{I}) = 1$. Therefore, if ϕ is valid, $\Pi(\phi) = 1 = 1 - N(\neg\phi)$. Otherwise, if ϕ is an unsatisfiable formula and the possibility distribution into consideration is normalised, then $\Pi(\phi) = 0 = 1 - N(\neg\phi)$. From these definitions, it follows that in possibilistic logic it is possible to have a case where $\Pi(\phi) = \Pi(\neg\phi) = 1$ or $N(\phi) = N(\neg\phi) = 0$, that means ignorance over ϕ.

As stated before, a possibilistic KB $\mathcal{H}$ defines a possibility distribution which in turn defines a preference over the interpretations. In fact, $\mathcal{H}$ defines a class of possibility distributions that are compliant with its constraints. Every formula $(\phi, \alpha) \in \mathcal{H}$ expresses

the constraint $N(\phi) \geq \alpha$.[1] We say that a possibility distribution π *satisfies* a possibilistic formula (ϕ, α) denoted $\pi \models (\phi, \alpha)$, iff the necessity measure $N_\pi(\phi)$ satisfies $N_\pi(\phi) \geq \alpha$.

A possibility distribution *satisfies* a KB $\mathcal{H}$ iff for every $(\phi, \alpha) \in \mathcal{H}$, π satisfies (ϕ, α), denoted $\pi \models \mathcal{H}$. A possibility distribution π_1 is more specific than a possibility distribution π_2 iff for every $\mathcal{I} \in \Omega$, $\pi_1(\mathcal{I}) \leq \pi_2(\mathcal{I})$. For a possibilistic KB $\mathcal{H}$ there may be multiple possibility distributions that satisfy it, since we are interested in modelling uncertainty we are interested in the least specific possibility distribution that satisfies $\mathcal{H}$. This means that we prefer the possibility distribution that assigns the highest possibility value to all elements in Ω while satisfying the constraints of $\mathcal{H}$. This respects the principle of minimum specificity [Dubois and Prade, 1987] that is in line with the intuition that the less it is known, the more different alternative events can be considered to be possible, in general. Analogously, the more it is known about a specific domain, the fewer possible events not complying with the current state of knowledge become. For instance, if I am somewhat certain that an elephant is a mammal, I should be more certain that an elephant is not an insect.

Given a possibilistic KB $\mathcal{H}$ and $\mathcal{I} \in \Omega$, we can define the least specific possibility distribution of $\mathcal{H}$ in this way:

$$\pi_{\mathcal{H}}(\mathcal{I}) = \begin{cases} 1 & \text{if for every } a \in \mathcal{H},\ \mathcal{I} \models a \\ 1 - \sup(\{\alpha \mid (\phi, \alpha) \in \mathcal{H} \text{ and } \mathcal{I} \models \neg\phi\}) & \text{otherwise.} \end{cases}$$

We denote by $\Pi_{\mathcal{H}}$ and $N_{\mathcal{H}}$ the possibility and necessity measures induced by $\pi_{\mathcal{H}}$, respectively. The least specific possibility distribution maximises the possibility values assigned to interpretations while satisfying the considered constraints. Proposition 2.5 formalises the previous statement and Example 2.6 illustrates this notion.

Proposition 2.5. *[Dubois et al., 1994] Let $\mathcal{H}$ be a possibilistic KB and let Ω be the set of all interpretations for the logic language $\mathfrak{L}$ considered. For any possibility distribution π, π satisfies $\mathcal{H}$ iff for every interpretation $\mathcal{I} \in \Omega$, we have that $\pi(\mathcal{I}) \leq \pi_{\mathcal{H}}(\mathcal{I})$.*

Example 2.6. Let $\mathfrak{L}$ be a propositional logic language defined on the set of variables $\mathsf{V} = \{\mathsf{v}_1, \mathsf{v}_2\}$. Assume $\mathcal{H}_{ex} = \{(\neg\mathsf{v}_1 \vee \neg\mathsf{v}_2, 0.2), (\mathsf{v}_1, 0.3), (\mathsf{v}_2, 0.8)\}$. The 0.3-cut of $\mathcal{H}_{ex}$, $\mathcal{H}_{ex,0.3}$ is $\{(\mathsf{v}_1, 0.3), (\mathsf{v}_2, 0.8)\}$ that is also equal to the 0.2-cut of $\mathcal{H}_{ex}$. The classical projection $\mathcal{H}^*_{ex}$ is $\{(\neg\mathsf{v}_1 \vee \neg\mathsf{v}_2), \mathsf{v}_1, \mathsf{v}_2\}$ and the classical projection with the 0.3-cut is $\mathcal{H}^*_{ex,0.3} = \{\mathsf{v}_1, \mathsf{v}_2\}$. The possibility distribution $\pi_{\mathcal{H}_{ex}}$ is computed with the (possibilistic)

[1] In this work, we are considering only "necessity valued possibilistic KBs", that is set of formulas that impose a lower bound on the necessity measure. The interested reader may read more about "generalised possibilistic logic" where formulas in the KB may impose also an upper bound on the possibility measure [Dubois et al., 1994].

v_1	v_2	$(\neg\mathsf{v}_1 \vee \neg\mathsf{v}_2, 0.2)$	$(\mathsf{v}_1, 0.3)$	$(\mathsf{v}_2, 0.8)$	$\pi_{\mathcal{H}_{ex}}$
1	1	0.8	1	1	0.8
1	0	1	1	0.2	0.2
0	1	1	0.7	1	0.7
0	0	1	0.7	0.2	0.2

Figure 2.2: Possibilistic truth table of $\mathcal{H}_{ex}$.

truth table in Figure 2.2. The interpretations that falsify the formula v_2 are associated with a low possibility degree because $\mathcal{H}_{ex}$ constraints v_2 to be almost certain (with necessity degree at least 0.8). The distribution $\pi_{\mathcal{H}_{ex}}$ is an example of a not normalised possibility distribution. ◁

Similarly to non-possibilistic logics, a possibilistic formula (ϕ, α) is a *logical consequence* of a KB $\mathcal{H}$ iff for any π satisfying $\mathcal{H}$, then π also satisfies (ϕ, α), written in symbols $\mathcal{H} \models (\phi, \alpha)$. By Proposition 2.7, it follows that computing the least specific possibility distribution is sufficient for checking if a possibilistic formula is entailed by the KB.

Proposition 2.7. *[Dubois and Prade, 1990a, Corollary 3.2.3] Let $\mathcal{H}$ be a possibilistic KB. For every possibilistic formula (ϕ, α), $\mathcal{H} \models (\phi, \alpha)$ iff $\pi_{\mathcal{H}} \models (\phi, \alpha)$.*

The possibility distribution $\pi_{\mathcal{H}_{ex}}$ induced by the KB $\mathcal{H}_{ex}$ in Example 2.6 entails the formula $(\mathsf{v}_1 \wedge \mathsf{v}_2, 0.3)$ because the possibility and necessity measure induced by $\pi_{\mathcal{H}_{ex}}$ give $N(\mathsf{v}_1 \wedge \mathsf{v}_2) = 1 - \Pi(\neg\mathsf{v}_1 \vee \neg\mathsf{v}_2) = 1 - \sup\{\pi_{\mathcal{H}_{ex}}(\mathcal{I}) \mid \mathcal{I} \models \neg\mathsf{v}_1 \vee \neg\mathsf{v}_2\} = 1 - 0.7 = 0.3$. By definition, $\pi_{\mathcal{H}_{ex}}$ satisfies $(\mathsf{v}_1 \wedge \mathsf{v}_2, 0.3)$ iff $N(\mathsf{v}_1 \wedge \mathsf{v}_2) \geq 0.3$. Since it is the case, $\pi_{\mathcal{H}_{ex}} \models (\mathsf{v}_1 \wedge \mathsf{v}_2, 0.3)$ and by Proposition 2.7, we have that $\mathcal{H}_{ex} \models (\mathsf{v}_1 \wedge \mathsf{v}_2, 0.3)$. $\pi_{\mathcal{H}_{ex}}$ satisfies also $(\mathsf{v}_1 \wedge \mathsf{v}_2, 0.2)$, $(\mathsf{v}_1 \wedge \mathsf{v}_2, 0.03)$ etc. (Proposition 2.8).

Proposition 2.8. *[Dubois and Prade, 1990a, Property 1 at page 453] Let $\mathcal{H}$ be a possibilistic KB. For every possibilistic formula (ϕ, α), if $\mathcal{H} \models (\phi, \alpha)$ then we have that $\mathcal{H} \models (\phi, \beta)$ for every β s.t. $0 < \beta \leq \alpha$.*

Clearly, we are interested in computing the highest valuation of an entailed formula. The deduction problem in possibilistic logic consists of identifying the highest valuation associated to a formula entailed by the possibilistic KB $\mathcal{H}$. Let ϕ be a classical formula. We denote by $\mathsf{val}(\phi, \mathcal{H}) = \sup\{\alpha \mid \mathcal{H} \models (\phi, \alpha)\}$.

Possibilistic logic is capable of managing inconsistent knowledge. If $\mathfrak{L}$ admits inconsistent KBs, the possibilistic extension $\mathfrak{L}_\pi$ admits *partial inconsistent* possibilistic KBs. The classical projection of the KB $\mathcal{H}_{ex}$ in Example 2.6 is inconsistent but not every formula can be entailed by $\mathcal{H}_{ex}$, for example $\mathcal{H}_{ex} \not\models (\mathsf{v}_1, 0.5)$. This is because possibilistic logic KBs allow for graded inconsistency. This degree is equal to the maximal value among the

smallest valuations associated to formulas that are involved in a contradiction in the KB. In Example 2.6, the KB $\mathcal{H}_{ex}$ is inconsistent up to degree 0.2 because $(\neg \mathsf{v}_1 \vee \neg \mathsf{v}_2, 0.2)$ is the formula with the smallest valuation present in every contradiction. A possibilistic KB $\mathcal{H}$ is (partially) inconsistent up to level α if it holds that $\mathcal{H} \models (\bot, \alpha)$ and its inconsistency degree is computed as follows:

$$\mathsf{inc}(\mathcal{H}) = \sup\{\alpha \mid \mathcal{H}^*_\alpha \text{ is inconsistent}\}.$$

The inconsistency level of $\mathcal{H} = \emptyset$ is 0 and for $\mathcal{H}_{ex}$ in Example 2.6, we have $inc(\mathcal{H}_{ex}) = 0.2$. Recall that we have chosen $\pi_{\mathcal{H}}$ to be the least possibility measure that satisfies $\mathcal{H}$. Another good reason for choosing the least specific distribution $\pi_{\mathcal{H}}$ as the characteristic measure for $\mathcal{H}$ is that it minimises the necessity measure N, i.e. it maximises the values of Π. As a consequence, $\pi_{\mathcal{H}}$ also minimises the inconsistency degree of $\mathcal{H}$.

It follows from the definition, that formulas non-trivially entailed must have a valuation greater than the inconsistency level of the KB. By Proposition 2.9, finding the valuation of a formula entailed by a possibilistic KB is reduced to finding the inconsistency degree of a formula. The complexity of reasoning in the possibilistic extension of $\mathfrak{L}$ increases only by a logarithmic factor on the number of valuations in the KB [Lang, 2000].

Proposition 2.9. *[Dubois and Prade, 1990a, Proposition 3.5.5] Let $\mathcal{H}$ be a possibilistic KB. For every possibilistic formula (ϕ, α), $\mathcal{H} \models (\phi, \alpha)$ iff $\mathcal{H} \cup \{(\neg\phi, 1)\} \models (\bot, \alpha)$ or equivalently $\mathsf{val}(\phi, \mathcal{H}) = Inc(\mathcal{H} \cup \{(\neg\phi, 1)\})$.*

Recall once again the KB $\mathcal{H}_{ex}$ in Example 2.6 and assume we would like to check what is $\mathsf{val}(\mathsf{v}_1 \wedge \mathsf{v}_2, \mathcal{H}_{ex})$. By Proposition 2.9 we compute $\mathsf{inc}(\mathcal{H}_{ex} \cup \{(\neg \mathsf{v}_1 \vee \neg \mathsf{v}_2, 1)\})$ that corresponds to checking for every valuation[2] α if $\mathcal{F} = (\mathcal{H}_{ex} \cup \{(\neg \mathsf{v}_1 \vee \neg \mathsf{v}_2, 1)\})^*_\alpha$ is inconsistent. Since $\mathcal{F}$ is a classical KB, this check is done using standard techniques.

2.4 Computational Learning Theory

In order to study learning tasks formally, we need an abstract learning framework that states precisely what are the objects that can be learned and what elements represent the source of information. For this reason, we introduce the notion of a *learning framework* $\mathfrak{F}$ that consists of a triple:

$$(\mathcal{E}, \mathcal{L}, \mu)$$

[2] Actually, clever approaches can be used, like binary search on the valuations of the KB.

where $\mathcal{E}$ is a non-empty set of *examples*, $\mathcal{L}$ is a non-empty *hypothesis space* and μ is a mapping function $\mu : \mathcal{L} \to 2^{\mathcal{E}}$ that maps each hypothesis in the hypothesis space to a set of examples [Konev et al., 2018]. Elements in $\mathcal{E}$ are statements or data that, together with the information provided by the function μ, characterise an abstract target the learner wants to learn. In this work, $\mathcal{L}$ contains an element $\mathcal{T}$ called *target* that the learner attempts to find. In literature, this is called the realisability assumption [Shalev-Shwartz and Ben-David, 2014] and in this work it means that we can represent the target concept in the same way we can represent the hypothesis.

Example 2.10. Let $\mathfrak{F} = (\mathcal{E}, \mathcal{L}, \mu)$ be a learning framework where $\mathcal{L}$ is a propositional logic language defined on the set of variables $\mathsf{V} = \{\mathsf{v}_1, \mathsf{v}_2\}$. $\mathcal{E}$ corresponds to the set of all formulas in $\mathcal{L}$ and μ relates propositional logic KBs $\mathcal{H} \in \mathcal{L}$ with formulas entailed by them, that is $\mu(\mathcal{H}) = \{\phi \in \mathcal{E} \mid \mathcal{H} \models \phi\}$. For the KB $\mathcal{H} = \{\neg\mathsf{v}_1 \vee \mathsf{v}_2, \mathsf{v}_1\}$, for instance, we have that $\mathsf{v}_1 \vee \mathsf{v}_2, \mathsf{v}_1, \mathsf{v}_2 \in \mu(\mathcal{H})$. ◁

We now define two *learning models* that state what the goal of the learning task is and what the allowed actions are for obtaining information and fulfil the goal.

The Exact Learning Model

We formally study learning of possibilistic logic KBs in the EL model [Angluin, 1988]. Given a learning framework $\mathfrak{F} = (\mathcal{E}, \mathcal{L}, \mu)$, the goal of this learning model is the exact identification of a *target* $\mathcal{T} \in \mathcal{L}$, by posing queries. Exact learning models the scenario where there is a learner and a teacher. The learner attempts to identify the unknown target by asking queries to the teacher who knows it. Each element $\mathcal{H}$ of $\mathcal{L}$ is assumed to be represented using a set of symbols $\Sigma_{\mathcal{H}}$ (the *signature* of $\mathcal{H}$ introduced in Section 2.1) that the learner knows. We say that $e \in \mathcal{E}$ is a *positive example* for $\mathcal{H} \in \mathcal{L}$ if $e \in \mu(\mathcal{H})$ and a *negative example* for $\mathcal{H}$ if $e \notin \mu(\mathcal{H})$. Given $\mathcal{T}, \mathcal{H} \in \mathcal{L}$, a *counterexample* for $\mathcal{T}$ and $\mathcal{H}$ is an example $e \in \mathcal{E}$ such that $e \in \mu(\mathcal{T}) \oplus \mu(\mathcal{H})$.[3] An example $e \in \mathcal{E}$ is a *positive counterexample* for $\mathcal{T}$ and $\mathcal{H}$ if $e \in \mu(\mathcal{T})$. It is a *negative counterexample* for $\mathcal{T}$ and $\mathcal{H}$ if $e \in \mu(\mathcal{H})$ otherwise. Example 2.11 clarifies these last definitions.

Example 2.11. Let $\mathfrak{F}$ be the propositional logic learning framework defined in Example 2.10 and let $\mathcal{H} = \{\neg\mathsf{v}_1 \vee \mathsf{v}_2, \mathsf{v}_1\}$ be a propositional logic KB. The formulas v_1 and $(\mathsf{v}_1 \wedge \mathsf{v}_2)$ in $\mathcal{E}$ are positive examples for $\mathcal{H}$ and the formulas $\neg\mathsf{v}_1$ and $(\mathsf{v}_2 \wedge \neg\mathsf{v}_1)$ are negative examples for $\mathcal{H}$. A counterexample for $\mathcal{H}$ and $\mathcal{L} = \{\neg\mathsf{v}_1\}$ may be $\neg\mathsf{v}_1$ because $\mathcal{L} \models \neg\mathsf{v}_1$ and $\mathcal{H} \not\models \neg\mathsf{v}_1$ or it may be v_1 because $\mathcal{H} \models \mathsf{v}_1$ and $\mathcal{L} \not\models \mathsf{v}_1$. ◁

[3] The symbol $\oplus$ denotes the symmetric difference of two sets A, B. That is $(A \setminus B) \cup (B \setminus A)$.

In this learning model, different communication protocols can be studied depending on which queries the learner is able to ask to the teacher. We consider four types of queries. The most studied communication protocol between the learner and the teacher supports two types of queries, *membership* and *equivalence* query. For any $\mathfrak{F} = (\mathcal{E}, \mathcal{L}, \mu)$ and target $\mathcal{T} \in \mathcal{L}$, we define the membership query oracle $\mathsf{MQ}_{\mathfrak{F},\mathcal{T}}$ and the equivalence query oracle $\mathsf{EQ}_{\mathfrak{F},\mathcal{T}}$. $\mathsf{MQ}_{\mathfrak{F},\mathcal{T}}$ is the oracle that takes as input some $e \in \mathcal{E}$ and returns 'yes' if $e \in \mu(\mathcal{T})$ and 'no' otherwise. A *membership query* is a call to the oracle $\mathsf{MQ}_{\mathfrak{F},\mathcal{T}}$. $\mathsf{EQ}_{\mathfrak{F},\mathcal{T}}$ is the oracle that takes as input a *hypothesis* $\mathcal{H} \in \mathcal{L}$ and returns 'yes' if $\mu(\mathcal{H}) = \mu(\mathcal{T})$ and a counterexample otherwise. There is no assumption regarding which counterexample is chosen by the oracle. An *equivalence query* is a call to $\mathsf{EQ}_{\mathfrak{F},\mathcal{T}}$.

Other queries we consider are *subset* and *superset* queries. Analogously, we define the subset query oracle $\mathsf{SbQ}_{\mathfrak{F},\mathcal{T}}$ and superset query oracle $\mathsf{SpQ}_{\mathfrak{F},\mathcal{T}}$. $\mathsf{SbQ}_{\mathfrak{F},\mathcal{T}}$ takes as input a hypothesis $\mathcal{H} \in \mathcal{L}$ and it returns 'yes' if $\mu(\mathcal{H}) \subseteq \mu(\mathcal{T})$ and a counterexample $e \in \mu(\mathcal{H}) \setminus \mu(\mathcal{T})$ otherwise. $\mathsf{SpQ}_{\mathfrak{F},\mathcal{T}}$ takes as input a hypothesis $\mathcal{H} \in \mathcal{L}$ and it returns 'yes' if $\mu(\mathcal{T}) \subseteq \mu(\mathcal{H})$ and a counterexample $e \in \mu(\mathcal{T}) \setminus \mu(\mathcal{H})$ otherwise.

In this work, we consider two main settings for concept learning. One in which the learner has to identify the target $\mathcal{T}$ with examples being formulas. The second where the examples are partial interpretations.

Given a learning framework $(\mathcal{E}, \mathcal{L}, \mu)$ with $\mathcal{L}$ being an FO logic language. When $\mathcal{E}$ is a set of formulas, and μ is the entailment relation ($\models$), we say that we are in the *learning from entailments* setting. Otherwise, we are in the *learning from (partial) interpretations* setting if $\mathcal{E}$ is a set of (partial) interpretations and μ maps KBs $\mathcal{H} \in \mathcal{L}$ to the set of (partial) interpretation (subset of $\mathcal{E}$) that satisfy $\mathcal{H}$.

Example 2.12. *The learning framework defined in Example 2.10 belongs to the learning from entailment setting. While the learning framework $\mathfrak{F} \coloneqq (\mathcal{E}, \mathcal{L}, \mu)$ where $\mathcal{E}$ is a set of partial interpretations, $\mathcal{L}$ is the propositional logic language, and $\mu(\mathcal{H} \in \mathcal{L}) = \{\mathcal{I} \in \mathcal{E} \mid \mathcal{I} \models \mathcal{H}\}$ belongs to the learning from interpretation setting.* ◁

Given a learning framework $\mathfrak{F} = (\mathcal{E}, \mathcal{L}, \mu)$, and a non-empty set of oracles Q, we say that $\mathfrak{F}$ is *exact learnable* with Q if there is an algorithm A such that for any $\mathcal{H} \in \mathcal{L}$:

- A can ask queries to any oracle in Q; and
- A always halts and outputs $\mathcal{H}' \in \mathcal{L}$ such that $\mathcal{H}' \equiv \mathcal{H}$.

Every learning framework $(\mathcal{E}, \mathcal{L}, \mu)$ is exact learnable if equivalence queries are available.

Algorithm 1: Exact Learning k-CNF formulas

Input: V: variables.
Output: A k-CNF formula equivalent to the target t.
Initialise $\mathcal{H}$ as the conjunction of every clause over V with at most k literals.
while $\mathsf{EQ}_{\mathfrak{F}_\pi,\mathcal{T}}(\mathcal{H})$ returns a counterexample $\mathcal{I}$ **do**
  for clause $\psi \in \mathcal{H}$ **do**
    if $\mathcal{I} \not\models \psi$ **then**
      $\mathcal{H} \leftarrow \mathcal{H} \setminus \{\psi\}$
    end if
  end for
end while
return $\mathcal{H}$

Theorem 2.13 ([Angluin, 1988])**.** *Let $\mathfrak{F} = (\mathcal{E}, \mathcal{L}, \mu)$ be an FO learning framework with $\mathcal{T} \in \mathcal{L}$, and Q a set of oracles such that $\mathsf{EQ}_{\mathfrak{F},\mathcal{T}} \in Q$. $\mathfrak{F}$ is exact learnable with Q.*

Proof. Let k be the string length of $\mathcal{T}$ with respect to $\mathcal{L}$. We can consider an algorithm A that enumerates all $\mathcal{H}^n \in \mathcal{L}$ such that $\mathcal{H}^n$ has string length $n \geq 1$ with respect to the language $\mathcal{L}$. For each of such a $\mathcal{H}^n$ (starting with $n = 1$), the learning algorithm A will call $\mathsf{EQ}_{\mathfrak{F},\mathcal{T}}$ with input $\mathcal{H}^n$. If the answer is 'yes', then A will halt and output $\mathcal{H}^n$. Otherwise, A will select another $\mathcal{H}^n_2 \in \mathcal{L}$ until all elements with length n in $\mathcal{L}$ have been queried to $\mathsf{EQ}_{\mathfrak{F},\mathcal{T}}$. After that, A will increase n by 1 and repeat this process until it has found a hypothesis equivalent to $\mathcal{T}$ ($n = k$). □

Otherwise, we can find learning frameworks that are not exact learnable with a specific combination of queries. For example, when only a membership query oracle is available.

Example 2.14. We define $\Phi_n := \exists x_1 \dots \exists x_n . \bigwedge_{i=0}^{n} r(x_i, x_{i+1})$ for $n \in \mathbb{N}^+$. Let $(\mathcal{E}, \mathcal{L}, \mu)$ be a learning framework where μ is the entailment relation and

$$\mathcal{E} := \{\Phi_n \mid n \in \mathbb{N}^+\} \quad \text{and} \quad \mathcal{L} := \{\exists x_0 \Phi_n \mid n \in \mathbb{N}^+\} \cup \{\forall x_0 \Phi_n \mid n \in \mathbb{N}^+\}.$$

So, we are in the learning from entailment setting. A learner may ask membership queries of the form Φ_n for an arbitrarily large n without being able to distinguish whether the target theory is $\exists x_0 \Phi_n$ or $\forall x_0 \Phi_n$. Knowing the signature of the target theory does not help the learner. ◁

To help familiarise with the notation and the learning process, we explain how a fragment of propositional logic called k-CNF can be exact learned with only equivalence queries in the learning from interpretations setting [Angluin, 1988]. A logic language $\mathfrak{L}$ is k-CNF if the number of literals in every clause in a formula $\phi \in \mathfrak{L}$ is at most k. We can run

an algorithm A that follows the steps in Algorithm 1. At first A generates a k-CNF formula $\mathcal{H}$ that is the conjunction of every clause over n variables ($|\mathsf{V}| = n$) with at most k literals. As there are in total at most $(2|\mathsf{V}| + 1)^k$ of such clauses, $\mathcal{H}$ has a polynomial size with respect to k and the number of literals $|\mathsf{V}|$. For an unknown target $\mathcal{T} \in \mathfrak{L}$, it always holds that $\mathcal{H} \models \mathcal{T}$, therefore A can only receive a counterexample $\mathcal{I}$ such that $\mathcal{I} \not\models \mathcal{H}$ and $\mathcal{I} \models \mathcal{T}$. As a consequence, A can delete all clauses $\psi \in \mathcal{H}$ satisfying $\mathcal{I} \not\models \psi$. Example 2.15 follows the steps of one such loop depicted in Algorithm 1.

Example 2.15. *Let $|\mathsf{V}| = 2$ and $k = 2$ and assume the target k-CNF formula is $\mathcal{T} \coloneqq (\mathsf{v}_1 \vee \mathsf{v}_2)$. Algorithm 1 will instantiate at first the formula $\mathcal{H}$:*

$$\mathsf{v}_1 \wedge \mathsf{v}_2 \wedge \neg\mathsf{v}_1 \wedge \neg\mathsf{v}_2 \wedge (\mathsf{v}_1 \vee \mathsf{v}_2) \wedge (\mathsf{v}_1 \vee \neg\mathsf{v}_2) \wedge (\neg\mathsf{v}_1 \vee \mathsf{v}_2) \wedge (\neg\mathsf{v}_1 \vee \neg\mathsf{v}_2).$$

After calling an equivalence query with input $\mathcal{H}$, the algorithm will receive a positive counterexample $\mathcal{I}$. Let such counterexample be $\mathcal{I} = \{(\mathsf{v}_1, \bot), (\mathsf{v}_2, \top)\}$. Therefore, the algorithm will refine its hypothesis $\mathcal{H}$ to:

$$\mathsf{v}_2 \wedge \neg\mathsf{v}_1 \wedge (\mathsf{v}_1 \vee \mathsf{v}_2) \wedge (\neg\mathsf{v}_1 \vee \mathsf{v}_2) \wedge (\neg\mathsf{v}_1 \vee \neg\mathsf{v}_2).$$

This ensures that the condition $\mathcal{I} \models \mathcal{H}$ is respected again. ◁

Afterwards, A can call the equivalence query again and repeat the same steps until the answer received is 'yes'. The maximum number of queries that the algorithm will ask is bounded by $(2|\mathsf{V}| + 1)^k$.

The PAC Learning Model

Requiring the learner to exactly identify an unknown concept may be a condition too demanding. Moreover, the assumption of oracles that answer queries may fail to hold in real world learning tasks. Therefore, in some scenarios it is more convenient to adopt a *passive* model that does not require an active interaction between the learning algorithm and the provider of information. The PAC learning model [Valiant, 1984] belongs to the class of learning models whose goal is to approximate a target concept from the observation of a *labelled* set of examples.

More precisely, let $\mathfrak{F} = (\mathcal{E}, \mathcal{L}, \mu)$ be a learning framework, where $\mathcal{E}$ is a set of examples, $\mathcal{L}$ is the hypothesis space, $\mu : \mathcal{L} \to \mathcal{E}$ is a surjective function, and $\mathcal{T} \in \mathcal{L}$ is the unknown target. We assume the existence of a probability distribution $\mathcal{D} : \mathcal{E} \to [0, 1]$ such that $\Sigma_{e \in \mathcal{E}} \mathcal{D}(e) = 1$. To keep the notation consistent with the EL model, we introduce the *example oracle* $\mathsf{EX}^{\mathcal{D}}_{\mathfrak{F},\mathcal{T}}$. It takes as input no argument, and when $\mathsf{EX}^{\mathcal{D}}_{\mathfrak{F},\mathcal{T}}$ is called, it draws

Algorithm 2: PAC Learning k-CNF formulas

Input: V: variables.
Output: A k-CNF formula.
Initialise $\mathcal{H}$ as the conjunction of every clause over V with at most k literals.
$i \leftarrow 1$
while there is a counterexample $\mathcal{I}$ in a sample of size $(1/\epsilon)(ln1/\delta + i\ ln2)$ from $\mathsf{EX}_{\mathfrak{F}_\pi,\mathcal{T}}(\mathcal{H})$ **do**
  for clause $\psi \in \mathcal{H}$ **do**
    if $\mathcal{I} \not\models \psi$ **then**
      $\mathcal{H} \leftarrow \mathcal{H} \setminus \{\psi\}$
    end if
  end for
  $i \leftarrow i + 1$
end while
return $\mathcal{H}$

an $e \in \mathcal{E}$ according to the probability distribution $\mathcal{D}$ and it returns the pair $(e, \mathbb{1}_{e\in\mu(\mathcal{T})})$ to the caller.[4] A *sample* generated by $\mathsf{EX}^{\mathcal{D}}_{\mathfrak{F},\mathcal{T}}$ is a (multi-)set of indexed classified examples, independently and identically distributed according to $\mathcal{D}$, and sampled by calling $\mathsf{EX}^{\mathcal{D}}_{\mathfrak{F},\mathcal{T}}$.

A learning framework $\mathfrak{F} = (\mathcal{E}, \mathcal{L}, \mu)$ is *PAC learnable* if there is a function $f : (0,1)^2 \to \mathbb{N}$ and a deterministic algorithm A such that, for every $\epsilon, \delta \in (0,1)$, every probability distribution $\mathcal{D}$ on $\mathcal{E}$, and every target $\mathcal{T} \in \mathcal{L}$ given a sample of size $m \geq f(\epsilon, \delta)$ generated by $\mathsf{EX}^{\mathcal{D}}_{\mathfrak{F},\mathcal{T}}$, the algorithm always halts and outputs $\mathcal{H} \in \mathcal{L}$ such that with probability at least $(1-\delta)$ over the choice of m examples in $\mathcal{E}$, we have that $\mathcal{D}(\mu(\mathcal{H}) \oplus \mu(\mathcal{T})) \leq \epsilon$.

There is a connection between the EL and PAC learning model due to the relation between equivalence and stochastic equivalence [Angluin, 1988]. Assume the learning framework $\mathfrak{F}$ is exact learnable with algorithm A. If instead of asking its i-th equivalence query, A generates a set of examples from the oracle $\mathsf{EX}_{\mathfrak{F},\mathcal{T}}$ of size

$$s_i \coloneqq \lceil \frac{1}{\epsilon}(ln\frac{1}{\delta} + i\ ln2) \rceil, \tag{2.1}$$

then the algorithm A can find a hypothesis with PAC guarantees. Indeed, when A generates samples with size s_i, the probability that A outputs a hypothesis $\mathcal{H}$ such that

[4] $\mathbb{1}_{e\in\mu(\mathcal{T})}$ is 1 if the condition $e \in \mu(\mathcal{T})$ is satisfied, and 0 otherwise.

$\mathcal{D}(\mu(\mathcal{H}) \oplus \mu(\mathcal{T})) > \epsilon$ is at most

$$\begin{aligned} \Sigma_{i=1}^{\infty}(1-\epsilon)^{s_i} &\leq \Sigma_{i=1}^{\infty} e^{\epsilon s_i} \\ &\leq \Sigma_{i=1}^{\infty} \frac{\delta}{2^i} \\ &\leq \delta \end{aligned} \tag{2.2}$$

Therefore, A will output a hypothesis compliant with the PAC learnability conditions if an equivalence query oracle is replaced by a example oracle, and if the size of the sample where the counterexamples are searched in is large enough (Equation (2.1)). Due to this connection, k-CNF formulas are also learnable in the PAC learning model. Algorithm 2 illustrates how to learn k-CNF in the PAC model. It is similar to Algorithm 1, but the algorithm has only access to an example oracle. The correctness follows according to Equation (2.2) and by the choice of the size of the sample in Equation (2.1).

2.5 Artificial Neural Networks

Acquiring information and generalising observed data is a challenging task to automate and implement in machines. A prominent method for gathering knowledge in the form of patterns from data is named *Artificial Neural Network* (ANN) [Goodfellow et al., 2016], whose intuition originally derivates from an analogy with the human brain.

Model Description

Mathematically, an ANN can be represented as a function $f : \mathbb{R}^n \to \mathbb{R}^m$ that takes as input a vector of size n and outputs a vector of predictions of size m. We denote vectors in bold with subscript to point to a specific value, that is $\boldsymbol{x}_i$ corresponds to the i-th element of the vector $\boldsymbol{x}$.

In this work, we will only consider a specific class of ANNs where the computation only flows forward from the input vector $\boldsymbol{x}$ to the output. For this reason, they are called *feedforward* neural networks, and they can be represented as a chain of functions $f(\boldsymbol{x}) = f^k(f^{k-1}(\cdots f^1(\boldsymbol{x})))$. Figure 2.3 shows an example where $k = 2$. In literature, when $k \geq 2$, we call it a *deep learning* neural network model. The length k of the chain defines the depth of the model which is always made of an *input layer*, the leftmost part of Figure 2.3, *hidden layers*, the central part where nodes are labelled with f^i and one *output layer*. The nodes f^i_j in the hidden layers are additional components of a single function f^i. They are called *hidden* because their output is shown neither in the input

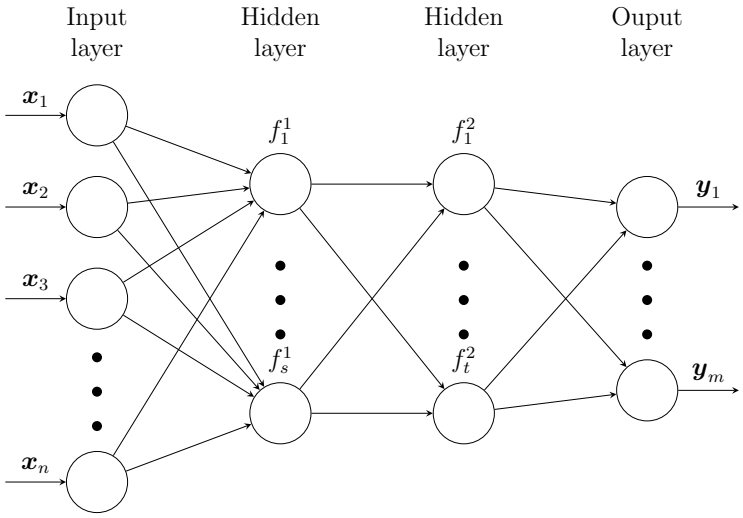

Figure 2.3: An example of an ANN with two hidden layers.

nor output data. The number of nodes per layer defines the *width* of the model.

Following the analogy of the human brain, each layer is assumed to extract increasingly complex patterns from the output of previous layers. Each node in Figure 2.3 represents an artificial neuron that can fire a signal depending on the received input signals. The abstraction capability of deep ANNs makes it efficient to automatically find custom representation of arbitrary datasets. The first layer can identify lines direction, the second layers contours, and next layers corners, shapes, etc. As a consequence, there can be little effort in finding a good representation of the data to be fed to the learning algorithm. For example, if we would like to have an image recognition system to classify cars and manually engineer a representation, we could think of how to define combinations of pixels that represent horizontal, vertical edges, etc. The abstraction of each layer makes ANNs a general algorithm for learning tasks.

Usually, the depth and width of the model is chosen a priori, and the challenge is to find an optimal configuration of the connections between each node from data. Each connection can be considered as a parameter of the ANN function that can strengthen or weaken its input signal. When needed, we explicitly write the parameters $\boldsymbol{w}$ of the ANN function with input $\boldsymbol{x}$ as $f(\boldsymbol{x}; \boldsymbol{w})$. As each layer in an ANN follows the same structure of $f(\boldsymbol{x}; \boldsymbol{w})$, we can also express each layer i as $f^i(\boldsymbol{x}; \boldsymbol{w}^i)$. Therefore:

$$f(\boldsymbol{x}; \boldsymbol{w}^1, \cdots, \boldsymbol{w}^k) \coloneqq f^k(f^{k-1}(\cdots f^1(\boldsymbol{x}; \boldsymbol{w}^1), \boldsymbol{w}^{k-1}), \boldsymbol{w}^k)$$

where each $\boldsymbol{w}^i$ is the parameter vector of layer i.

Algorithm 3: The *Stochastic Gradient Descend* (SGD) optimisation algorithm.

Input: parameters: $\boldsymbol{w}$, error function: $\mathbb{J}_{\mathsf{D}}$, learning rate: α, iterations: epochs.
Output: updated parameters $\boldsymbol{w}$.
count $\leftarrow 0$
while count $<$ epochs **do**
  Shuffle D into s random samples D_i, $1 \leq i \leq s$
  for $1 \leq i \leq s$ **do**
    for $1 \leq j \leq n$ **do**
      $\boldsymbol{w}_j \leftarrow \boldsymbol{w}_j - \alpha \frac{\partial \mathbb{J}_{\mathsf{D}_i}(\boldsymbol{w})}{\boldsymbol{w}_j}$
    end for
  end for
end while
return $\boldsymbol{w}$

Optimisation

Identifying the optimal value for each parameter in an ANN is a computationally complex problem [Baskakov and Arseniev, 2021]. But, experimental evaluations have shown that in practice optimisation procedures based on *gradient descent* have nice guarantees of finding solutions with good-performing models. These optimisation algorithms require a score of the learned model with respect to the true label of any input vector $\boldsymbol{x}$ of a given dataset D. For instance, let f^* be the unknown target function that we would like to approximate. A score of correctness of the model f can be computed with the *Mean Squared Error* (MSE) function:

$$\mathbb{J}_{\mathsf{D}}(\boldsymbol{w}) \coloneqq \Sigma_{\boldsymbol{x} \in \mathsf{D}} (f^*(\boldsymbol{x}) - f(\boldsymbol{x}; \boldsymbol{w}))^2 \tag{2.3}$$

The closer to 0, the more the model $f(\cdot; \boldsymbol{w})$ approximates the target. Therefore, the training process assumes that the true classification labels of input vectors $\boldsymbol{x}$ are available while optimising the parameters in $\boldsymbol{w}$. We can imagine the MSE function in a $|\boldsymbol{w}|$-dimensional space where the minimums (or minimum if there is only one global minimum) represent an optimal configuration of the parameters $\boldsymbol{w}$.

Gradient-descent learning methods can exploit the continuity of the MSE function and gradually update the values in $\boldsymbol{w}$ following the slope of the curve. This is achieved by computing the first-order derivative, denoted $\frac{\partial \mathbb{J}_{\mathsf{D}}(\boldsymbol{w})}{\boldsymbol{w}_j}$, of $\mathbb{J}_{\mathsf{D}}(\boldsymbol{w})$ with respect to every parameter $\boldsymbol{w}_j$ and decreasing the value of $\boldsymbol{w}_j$ if $\frac{\partial \mathbb{J}_{\mathsf{D}}(\boldsymbol{w})}{\boldsymbol{w}_j} > 0$, otherwise $\boldsymbol{w}_j$ is increased. There are many (first-order-based) optimisation algorithms based on this idea like SGD Bottou [2004]. The step-size update value to add or subtract to the parameters is regulated by the *learning rate* $\alpha \in \mathbb{R}^+$, usually less than 1. Implementations also decrease the value α as the algorithm loops to guarantee converges to the local minimum.

Chapter 3

RIDDLE: Rule Induction with Deep Learning

Chemical, oil [Bratko, 1993], energy companies [WJ., 1987] profit from the adoption of rule induction algorithms. Also, the advantage of finding interpretable patterns from data, benefits the field of medicine [Podgorelec et al., 2002; Scala et al., 2019], engineering (fault detection) [Dhanraj et al., 2022], and frauds [Xu et al., 2018], to name a few. These algorithms express patterns found in data in the form of associative ('if-then') *rules* [Cohen, 1995; Dash et al., 2018; Kusters et al., 2022; Vreeken et al., 2011] effectively aiding users in decision-making.

One main limitation for the adoption of rule induction approaches is that they have to solve hard discrete combinatorial problems. The search space for rule induction algorithms is often discrete and grows exponentially with respect to the number of the symbols available for expressing the patterns [Cohen, 1995; Dash et al., 2018; Hühn and Hüllermeier, 2009]. Hence, their scalability suffers when compared to classification methods that can rely on techniques tailored for the optimisation of differentiable functions, such as deep learning with gradient-based optimisation [Elkano et al., 2020; Kusters et al., 2022]. Indeed, deep learning approaches have excelled in many tasks, including classification [Druzhkov and Kustikova, 2016], time series [Torres et al., 2021] image segmentation [Minaee et al., 2021], text and image generation [Fatima et al., 2022; Ramesh et al., 2021], etc. The success of deep learning approaches is due to the flexibility to handle many different forms of data and scalability regarding the current technology on hardware. Unfortunately, the most successful trained deep learning models are not interpretable. Therefore, the patterns found during training phase cannot be inspected and tested to give support for high-stakes decisions [Dash et al., 2018; Rudin, 2019].

In this chapter, we propose a novel *Artificial Neural Network* (ANN) architecture dubbed RIDDLE: **R**ule **I**n**D**uction with **D**eep **LE**arning. This architecture is based on the framework of possibility theory. As a consequence, learned rules are supplied with a confidence value that states how certain the model is about the respective rule (necessity degree). RIDDLE can find patterns expressible in full possibilistic propositional language. More precisely, in the possibilistic extension of *Conjunctive Normal Form* (CNF).

RIDDLE has two main advantages over most traditional rule induction methods:

- It does not have 'sharp' decision boundaries. That is, the learning task can be cast into a differentiable error function. Similarly to differentiable inductive logic programming techniques and further developments [Shindo et al., 2021]. So, we can employ efficient optimisation algorithms for tuning the parameters of the model.
- The order of the learned rules is irrelevant, that is, they yield rule sets instead of lists [Hühn and Hüllermeier, 2009]. As a consequence the process of learning many rules can be parallelised.

We test the performance of RIDDLE with 15 benchmark datasets, spanning from the medical domain to finance. We show that our method has state-of-the-art performance (accuracy) compared with other established rule induction algorithms. We noticed that RIDDLE shines especially on datasets with uncertain information.

In Section 3.1, we describe the RIDDLE architecture and in Section 3.2, we provide an empirical comparison between RIDDLE and FURIA [Hühn and Hüllermeier, 2009], a prominent fuzzy rule induction algorithm. We mention future steps in Section 3.3.

3.1 Introducing RIDDLE

Let V be the set of propositional variable in our domain, that is we represent rules with symbols from V. For the rest of this chapter, we assume an arbitrary but fixed ordering of the variables in V: $(\mathsf{v}_1, \ldots, \mathsf{v}_n, \mathsf{t}_1, \ldots, \mathsf{t}_m)$. We want to predict the certainty degree of the variable t_j, for $1 \leq j \leq m$, with the information provided by v_i for $1 \leq i \leq n$. We omit the subscript j from t_j to denote just one of the many target variables. More in detail, our goal is to predict $\Pi(\neg\mathsf{t})$ (or $\Pi(\mathsf{t})$), of a target variable t, from the possibility degrees of variables $\mathsf{v}_1, \neg\mathsf{v}_1, \ldots, \mathsf{v}_n, \neg\mathsf{v}_n$, and compute how necessary the target variable is according to our input with $N(\mathsf{t}) = 1 - \Pi(\neg\mathsf{t})$ (or $N(\neg\mathsf{t})$). Therefore, with the RIDDLE architecture, an ANN can be seen as a function $N : [0,1]^{2n} \to [0,1]^{2m}$ that takes as

input a vector of possibility values associated to $\mathsf{v}_1, \neg\mathsf{v}_1, \ldots, \mathsf{v}_n, \neg\mathsf{v}_n$ and outputs the possibilities of each target literal $\mathsf{t}_1, \neg\mathsf{t}_1, \ldots, \mathsf{t}_m, \neg\mathsf{t}_m$ (Remark 3.1).

Remark 3.1. *In practice, we may be interested in predicting the value of only one between t_j and $\neg\mathsf{t}_j$. In this case N can be defined as $N : [0,1]^{2n} \to [0,1]^s$ with $s \leq 2m$. Such that the output of N is a vector corresponding to the possibility value of only s literals among $\mathsf{t}_1, \neg\mathsf{t}_1, \ldots, \mathsf{t}_m, \neg\mathsf{t}_m$.*

We would like to train a N on dataset of labelled examples. For this reason, we assume a general setting in which the dataset can be represented as a set of (partial) interpretations $\mathrm{I} \coloneqq \{\mathcal{I}_1, \ldots, \mathcal{I}_d\}$ where some rules r_j with $1 \leq j \leq k$ of the form $(\phi \to \mathsf{t})$ hold. That is for each $1 \leq i \leq d$ and $1 \leq j \leq k$, we have $\mathcal{I}_i \models \mathsf{r}_j$. We can convert the statements expressed by the partial interpretations $\mathcal{I} \in \mathrm{I}$ to possibilistic degrees via the method proposed by Joslyn [1991] to estimate possibilities from imprecise data. To do so, we first define the set

$$\mathrm{I}_{\mathcal{I}} \coloneqq \{\mathcal{I}' \in \mathrm{I} \mid \forall \mathsf{v} \in \mathsf{V}, \mathcal{I}(\mathsf{v}) = ? \text{ or } \mathcal{I}(\mathsf{v}) = \mathcal{I}'(\mathsf{v})\}$$

that contains precisely the interpretations in I that differ only on the unknown values of $\mathcal{I}$. Then, from $\mathrm{I}_{\mathcal{I}}$, we count the number of interpretations that satisfy a literal l which is given by $\mathsf{c}_{\mathrm{I}_{\mathcal{I}}}(l) \coloneqq |\{\mathcal{I} \in \mathrm{I}_{\mathcal{I}} \mid \mathcal{I} \models l\}|$. Finally, the possibility associated to a literal l according to the facts in $\mathcal{I}$ and I, is defined as

$$\Pi^{\mathrm{I}_{\mathcal{I}}}(l) \coloneqq \mathsf{c}_{\mathrm{I}_{\mathcal{I}}}(l) / \max(\mathsf{c}_{\mathrm{I}_{\mathcal{I}}}(l), \mathsf{c}_{\mathrm{I}_{\mathcal{I}}}(\neg l)).$$

Therefore, from I we can get the set of possibility degrees for each input literal $\mathsf{D} \coloneqq \{\boldsymbol{x}^1, \ldots, \boldsymbol{x}^d\}$ where for any $\mathcal{I}_i \in \mathrm{I}$,

$$\boldsymbol{x}^i \coloneqq (\Pi^{\mathrm{I}_{\mathcal{I}_i}}(\mathsf{v}), \Pi^{\mathrm{I}_{\mathcal{I}_i}}(\neg\mathsf{v}_1), \ldots, \Pi^{\mathrm{I}_{\mathcal{I}_i}}(\mathsf{v}_n), \Pi^{\mathrm{I}_{\mathcal{I}_i}}(\neg\mathsf{v}_n)).$$

The number j in $\boldsymbol{x}^i_j$ corresponds to the value at the j-th position. We denote the associated formula with respect to $\mathcal{I}_i \in \mathrm{I}$ by

$$\mathcal{X}^i \coloneqq \{(l, 1 - \Pi^{\mathrm{I}_{\mathcal{I}_i}}(\neg l)) \mid \Pi^{\mathrm{I}_{\mathcal{I}_i}}(\neg l) < 1\}.$$

Example 3.2 provides a clarification for these definitions.

Example 3.2. *Let* $\mathrm{I} = \{\mathcal{I}_1, \mathcal{I}_2\}$ *with*

$$\mathcal{I}_1 = \{(\mathsf{v}_1, \top), (\mathsf{v}_2, \top), (\mathsf{t}, \top)\}$$
$$\mathcal{I}_2 = \{(\mathsf{v}_1, \top), (\mathsf{v}_2, ?), (\mathsf{t}, ?)\}.$$

We have $\boldsymbol{x}^1 = (1, 0, 1, 0, 1, 0)$*, and* $\boldsymbol{x}^2 = (1, 0, 1, 1/2, 1, 1/2)$*.* $\boldsymbol{x}^2_4$ *is* $1/2$ *and* $\mathcal{X}^2 \coloneqq \{(\mathsf{v}_1, 1), (\mathsf{v}_2, 1/2), (\mathsf{t}, 1/2)\}$*.* ◁

To properly define the architecture later in this section, and make use of a differentiable error function, we employ the function log-sum-exp:

$$\mathsf{LSE}_\alpha(x_1, \ldots, x_n) \coloneqq \frac{1}{\alpha} \ln(e^{\alpha x_1} + \cdots + e^{\alpha x_n}),$$

as a smooth approximation of the max (min) function when $\alpha \to \infty$ ($\alpha \to -\infty$). We write LSE_{max} for LSE_{30} and LSE_{min} for LSE_{-30} to visually aid the reader in the next section and because we use LSE_{max} and LSE_{min} in our implementation. The number 30 is chosen empirically as a good parameter for the smooth approximation.

Theoretical Motivation

With V: $(\mathsf{v}_1, \ldots, \mathsf{v}_n, \mathsf{t})$, let $\mathcal{T} = \{(\phi_i \to \mathsf{t}, \alpha_i) \mid 1 \leq i \leq k\}$, with $N_\mathcal{T}(\mathsf{t}) = 0$, be the target set of possibilistic rules. We would like to identify rules in $\mathcal{T}$ having at our disposal the dataset $\mathsf{D} = \{\boldsymbol{x}^1, \cdots, \boldsymbol{x}^d\}$ of possibility degrees over variables in V that respect the possibility values conditions imposed by rules in $\mathcal{T}$. Under this setting we can prove an important property of the possibility measure with Lemma 3.3.

Lemma 3.3. *Given* $\mathcal{T} = \{(\phi_i \to \mathsf{t}, \alpha_i) \mid 1 \leq i \leq k\}$*, with* $N_\mathcal{T}(\mathsf{t}) = 0$*, and a set of possibilistic literals* $\mathcal{K} = \{(l_j, \alpha_j) \mid 1 \leq j \leq u\}$*, if we set* $\mathcal{F} = \mathcal{T} \cup \mathcal{K}$*, then we have*

$$\Pi_\mathcal{F}(\neg\mathsf{t}) = \min\{\max\{1 - \alpha_i, \Pi_\mathcal{F}(\neg\phi_i) \mid 1 \leq i \leq k\}\}.$$

Proof. By Proposition 2.9 and $N_\mathcal{T}(\mathsf{t}) = 0$, we obtain that

$$\begin{aligned} N_\mathcal{F}(\mathsf{t}) &= \max\{\alpha \mid \mathcal{F} \cup \{(\neg\mathsf{t}, 1) \models (\bot, \alpha)\} \\ &= \max\{N_\mathcal{F}(\phi_i \wedge (\phi_i \to \mathsf{t})) \mid 1 \leq i \leq k\}. \end{aligned}$$

From **Minitivity**, we get $N_\mathcal{F}(\mathsf{t}) = \max\{\min(N_\mathcal{F}(\phi_i), \alpha_i) \mid 1 \leq i \leq k\}$. Moreover, For any $\boldsymbol{x} \in [0, 1]^n$, it holds $(1 - \max(\boldsymbol{x})) = \min(1 - \boldsymbol{x})$, hence by the relation between possibility and necessity:[1]

$$\begin{aligned} \Pi_\mathcal{F}(\neg\mathsf{t}) &= 1 - \max\{\min(N_\mathcal{F}(\phi_i), \alpha_i) \mid 1 \leq i \leq k\} \\ &= \min\{\max\{1 - \alpha_i, \Pi_\mathcal{F}(\neg\phi_i) \mid 1 \leq i \leq k\}. \end{aligned}$$

□

[1] $\Pi(\phi) = 1 - N(\neg\phi)$.

For convenience, we denote by $\psi_i \coloneqq l_{i,1} \vee \ldots \vee l_{i,s_i}$ the clause $\neg\phi_i$ for any rule $(\phi_i \rightarrow \mathsf{t}, \alpha_i) \in \mathcal{T}$. By Lemma 3.3 and **Maxitivity**, for any formula $\mathcal{X}$ we can compute $\Pi_{\mathcal{T}\cup\mathcal{X}}(\neg\mathsf{t})$ as

$$\begin{aligned}\Pi_{\mathcal{T}\cup\mathcal{X}}(\neg\mathsf{t}) = &\min\{\max\{1-\alpha_i, \Pi_{\mathcal{T}\cup\mathcal{X}}(\psi_i) \mid 1 \leq i \leq k\}\} \\ &\text{with } \Pi_{\mathcal{T}\cup\mathcal{X}}(\psi_i) = \max(\Pi_{\mathcal{T}\cup\mathcal{X}}(l_{1,i}), \ldots, \Pi_{\mathcal{T}\cup\mathcal{X}}(l_{1,s_i})).\end{aligned} \tag{3.1}$$

By definition of $\mathcal{T}$, $\Pi_{\mathcal{T}\cup\mathcal{X}}(\psi_i)$ is equal to $\Pi_{\mathcal{X}}(\psi_i)$. Therefore, we can propagate the known uncertainty of input variables with $\boldsymbol{x}$ to obtain the certainty degrees of the unknown target variable t with min-max operations. In practice, we do not know what rules $(\phi_i \rightarrow \mathsf{t})$ hold in $\mathcal{T}$ and their necessity degree α_i. But, such rules constrain every possibility degree in D that we can use to induce ϕ_i and α_i, with $1 \leq i \leq k$. Now, we describe RIDDLE, a novel neural network architecture tailored for rule induction leveraging the uncertainty propagation properties of a possibility measure.

Architecture

We can alternatively compute $\Pi_{\mathcal{T}\cup\mathcal{X}}(\psi_i)$ as a parametrised combination of product and maximum operators:

$$\Pi_{\mathcal{T}\cup\mathcal{X}}(\psi_i) = \Pi_{\mathcal{X}}(\psi_i) = \max(\boldsymbol{w}_1^{\psi_i}\Pi_{\mathcal{X}}(\mathsf{v}_1), \boldsymbol{w}_2^{\psi_i}\Pi_{\mathcal{X}}(\neg\mathsf{v}_1), \ldots, \boldsymbol{w}_{2n}^{\psi_i}\Pi_{\mathcal{X}}(\neg\mathsf{v}_n))$$

where for odd (even) $1 \leq t \leq n$, $\boldsymbol{w}_t^{\psi_i} \in [0,1]$ selects to what degree v_t $(\neg\mathsf{v}_t)$ appears in ψ_i. In matrix notation with input $\boldsymbol{x} \in \mathsf{D}$, this operation becomes

$$f_\psi(\boldsymbol{x}) \coloneqq \boldsymbol{x} \star \boldsymbol{w}^\psi = \boldsymbol{x} \star [\boldsymbol{w}_1^\psi, \boldsymbol{w}_2^\psi, \ldots, \boldsymbol{w}_{2n}^\psi]^T = \Pi_{\mathcal{X}}(\psi_i),$$

where $\star$ denotes the matrix dot product with the sum replaced by the LSE_{max} operator. Lemma 3.4 formally states that for any clause ψ, we can find $f_\psi : [0,1]^n \rightarrow [0,1]$ such that $\Pi_{\mathcal{X}}(\psi) = f_\psi(\boldsymbol{x})$.

Lemma 3.4. *For any clause ψ and formula $\mathcal{X}$ over V, there is a vector $\boldsymbol{w}^\psi \in [0,1]^{|\mathsf{V}|}$, such that $f_\psi(\boldsymbol{x}) = \Pi_{\mathcal{X}}(\psi)$.*

Proof. Let $\psi \coloneqq l_1 \vee \cdots \vee l_s$. By the **Maxitivity** property,[2] we have

$$\Pi_{\mathcal{X}}(\psi) = \max(\Pi_{\mathcal{X}}(l_1), \ldots, \Pi_{\mathcal{X}}(l_s)) = f_\psi(\boldsymbol{x}) = \boldsymbol{x} \star \boldsymbol{w}^\psi$$

[2] $\Pi(\psi_1 \vee \psi_2) = \max(\Pi(\psi_1), \Pi(\psi_2))$.

We can assign for odd $1 \leq t \leq n$, the value $\boldsymbol{w}_t^{\psi_i} = 1$ ($\boldsymbol{w}_{2t}^{\psi_i} = 1$) if v_t ($\neg\mathsf{v}_t$) appears as a top-level literal in the disjunct ψ_i, otherwise the value 0. By definition, we get $\max(\Pi_{\mathcal{X}}(l_1), \ldots, \Pi_{\mathcal{X}}(l_s)) = f_\psi(\boldsymbol{x})$. □

As a consequence, we can approximate the computation of the value $\Pi_{\mathcal{T}\cup\mathcal{X}}(\neg\mathsf{t})$ in Equation (3.1) with the respective smooth approximations:

$$\begin{aligned}\Pi_{\mathcal{T}\cup\mathcal{X}}(\neg\mathsf{t}) &= \mathsf{LSE}_{min}(\mathsf{LSE}_{max}(\boldsymbol{\beta}, \boldsymbol{x} \star [\boldsymbol{w}^{\psi_1}, \cdots, \boldsymbol{w}^{\psi_k}])) \\ &= \mathsf{LSE}_{min}(\mathsf{LSE}_{max}(\boldsymbol{\beta}_i, f_{\psi_i}(\boldsymbol{x}) \mid 1 \leq i \leq k)).\end{aligned} \tag{3.2}$$

The vector $\boldsymbol{\beta} \in [0,1]^k$ is the parameter that approximates $1 - \alpha_i$. Therefore, the (possibilistic) rule induction problem of finding rules in $\mathcal{T}$ is reduced to selecting the right value for each parameter $\boldsymbol{w}^{\psi_i}$ and $\boldsymbol{\beta}$.

We can improve this method by exploiting the associative property of the LSE_{max} operator and compute $f_\psi(\boldsymbol{x})$ as $\mathsf{LSE}_{max}(f_{\Psi_1}(\boldsymbol{x}), \ldots, f_{\Psi_l}(\boldsymbol{x}))$, where each Ψ_j with $1 \leq j \leq l$ is a subformula of ψ. Example 3.5 clarifies the idea behind it. Additionally, some rule antecedents ϕ_i, ϕ_j (with $i \neq j$) in the target $\mathcal{T}$ may share common subformulas. So, by representing subformulas, we can decrease the number of parameters required to represent a clause ψ' with $f_{\psi'}(\boldsymbol{x})$ by stratifying $f_{\psi'}(\boldsymbol{x})$.

Example 3.5. *Given* $\psi_1 \coloneqq \mathsf{v}_1 \vee \neg\mathsf{v}_2 \vee \mathsf{v}_3$, $\psi_2 \coloneqq \mathsf{v}_1 \vee \neg\mathsf{v}_2 \vee \mathsf{v}_4$, *and possibilities* $\boldsymbol{x} \in [0,1]^8$, *we can compute*

$$\begin{aligned} f_{\mathsf{v}_1\vee\neg\mathsf{v}_2}(\boldsymbol{x}) &= \boldsymbol{x} \star \boldsymbol{w}^{\mathsf{v}_1\vee\neg\mathsf{v}_2}, \\ f_{\mathsf{v}_3}(\boldsymbol{x}) &= \boldsymbol{x} \star \boldsymbol{w}^{\mathsf{v}_3}, \text{ and} \\ f_{\mathsf{v}_4}(\boldsymbol{x}) &= \boldsymbol{x} \star \boldsymbol{w}^{\mathsf{v}_4} \end{aligned}$$

at first. Then, we can compute

$$\begin{aligned} f_{\psi_1}(\boldsymbol{x}) &= ([f_{\mathsf{v}_1\vee\neg\mathsf{v}_2}(\boldsymbol{x}), f_{\mathsf{v}_3}(\boldsymbol{x}), f_{\mathsf{v}_4}(\boldsymbol{x})] \star [1,1,0]^T) \\ f_{\psi_2}(\boldsymbol{x}) &= ([f_{\mathsf{v}_1\vee\neg\mathsf{v}_2}(\boldsymbol{x}), f_{\mathsf{v}_3}(\boldsymbol{x}), f_{\mathsf{v}_4}(\boldsymbol{x})] \star [1,0,1]^T) \end{aligned}$$

The number of parameters needed to represent clauses increases with the number of clauses. If the clauses to represent have many common clauses, representing them first can be beneficial. ◁

Figure 3.1 depicts a two hidden layers ANN with an architecture based on the idea of identifying subformulas. For an $l \geq 0$, and $i \geq 1$, let $\mathsf{HL}_i^l : [0,1]^{\mathfrak{I}_l} \to [0,1]$ be the function

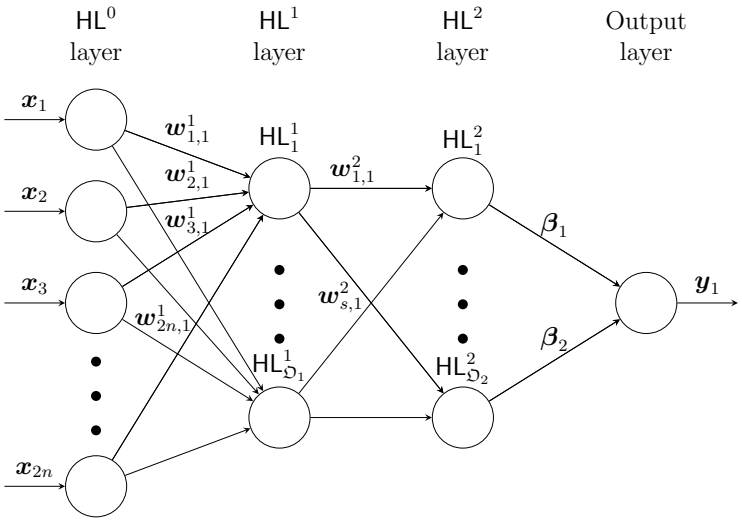

Figure 3.1: An example of an ANN with RIDDLE architecture and one output node.

that takes as input an example $\boldsymbol{x} \in \mathsf{D}$ ($\mathfrak{I}_l$ possibility values) and computes $f_{\psi_i}(\boldsymbol{x})$ with:

$$\mathsf{HL}_i^l(\boldsymbol{x}) \coloneqq \mathsf{LSE}_{max}(\boldsymbol{w}_{i,j}^l \mathsf{HL}_j^{l-1}(\boldsymbol{x}) \mid 1 \leq j \leq \mathfrak{I}_l)$$
$$\mathsf{HL}_i^0(\boldsymbol{x}) \coloneqq \boldsymbol{x}_i,$$

where for each $1 \leq h \leq l$, the value of every element in the matrix $\boldsymbol{w}^h$ is in the interval $[0, 1]$. In other words, for $1 \leq h \leq l$ the function $\mathsf{HL}_i^h(\boldsymbol{x})$ computes $\Pi_{\mathcal{X}}(\Psi)$ for a subformula Ψ of a clause ψ, in the same way $f_\psi(\boldsymbol{x})$ computes $\Pi_{\mathcal{X}}(\psi_i)$. For a fixed $l \geq 1$ and $i \geq 1$, HL_i^l can be considered as node i in the l-th layer in the neural network structure (Figure 3.1). Therefore, each layer $1 \leq h \leq l$, is a function $\mathsf{HL}^h : [0,1]^{\mathfrak{I}_h} \to [0,1]^{\mathfrak{O}_h}$, with $\mathfrak{I}_h, \mathfrak{O}_h \geq 1$ freely chosen (hyperparameters) that obey the constraint posed by the standard matrix dot-product. Finally, we can define a RIDDLE model (with one output) that takes as input $\boldsymbol{x} \in \mathsf{D}$ and outputs the possibility of a target literal as

$$\mathsf{RIDDLE}(\boldsymbol{x}) = \mathsf{LSE}_{min}(\mathsf{LSE}_{max}(\boldsymbol{\beta_i}, \mathsf{HL}_i^l(\boldsymbol{x}) \mid 1 \leq i \leq k)). \tag{3.3}$$

where $k \geq 1$ is a hyper parameter that defines the number of nodes in the last layer.

Theorem 3.6 shows that the defined architecture behaves as expected.

Theorem 3.6. *Given $\mathcal{T} = \{(\phi_i \to \mathsf{t}, \alpha_i) \mid 1 \leq i \leq k\}$ with $N_{\mathcal{T}}(\mathsf{t}) = 0$, we can find a configuration of the parameters in* RIDDLE *such that for any $\mathcal{X} = \{(l_j, \alpha_j) \mid 1 \leq j \leq u\}$, we have $\mathsf{RIDDLE}(\boldsymbol{x}) = \Pi_{\mathcal{T} \cup \mathcal{X}}(\neg \mathsf{t})$.*

Proof. For all literals l, by definition $N_{\mathcal{T}}(l) = 0$. Hence, for any clause ψ, $\Pi_{\mathcal{T}\cup\mathcal{X}}(\psi) = \Pi_{\mathcal{X}}(\psi)$. By Lemma 3.4 and associativity of max, we can set the values of the parameters in $\mathsf{HL}_i^l(\boldsymbol{x})$ so that it computes $\Pi_{\mathcal{X}}(\phi_i)$. By Equations (3.2) and (3.3), we get that $\mathsf{RIDDLE}(\boldsymbol{x})$ computes $\Pi_{\mathcal{T}\cup\mathcal{X}}(\neg\mathsf{t})$ like in Equation (3.1). □

The ideas developed so far can be trivially generalised to the case with many target variables, that is when V: $(\mathsf{v}_1, \ldots, \mathsf{v}_n, \mathsf{t}_1, \ldots, \mathsf{t}_m)$. The architecture is represented as:

$$\begin{aligned}\mathsf{RIDDLE}(\boldsymbol{x}) = (&\mathsf{LSE}^1_{min}(\mathsf{LSE}_{max}(\boldsymbol{\beta_i}^1, \mathsf{HL}_i^l(\boldsymbol{x}) \mid 1 \le i \le k)),\\ &\cdots,\\ &\mathsf{LSE}^s_{min}(\mathsf{LSE}_{max}(\boldsymbol{\beta_i}^s, \mathsf{HL}_i^l(\boldsymbol{x}) \mid 1 \le i \le k))),\end{aligned} \tag{3.4}$$

with $s \le 2m$ as explained in Remark 3.1. A direct consequence of Equation (3.4) and Theorem 3.6 is the following.

Corollary 3.7. *Given* $\mathsf{V} = \{\mathsf{v}_1, \ldots, \mathsf{v}_n, \mathsf{t}_1, \ldots, \mathsf{t}_m\}$ *and* $\mathcal{T} = \{(\phi_i \to \mathsf{t}, \alpha_i) \mid 1 \le i \le k, \mathsf{t} \in \{\mathsf{t}_1, \ldots, \mathsf{t}_m\}\}$*, with* $N_{\mathcal{T}}(\mathsf{t}) = 0$*, we can find a configurations of the parameters in* RIDDLE *such that for any set of literals* $\mathcal{X} = \{(l_j, \alpha_j) \mid 1 \le j \le s\}$*, we have*

$$\mathsf{RIDDLE}(\boldsymbol{x}) = (\Pi_{\mathcal{T}\cup\mathcal{X}}(\mathsf{t}_1), \Pi_{\mathcal{T}\cup\mathcal{X}}(\neg\mathsf{t}_1), \cdots, \Pi_{\mathcal{T}\cup\mathcal{X}}(\mathsf{t}_m), \Pi_{\mathcal{T}\cup\mathcal{X}}(\neg\mathsf{t}_m)).$$

Theorem 3.6 (or Corollary 3.7) relies on Lemma 3.4 which requires all parameters to be in $[0, 1]$. Thus, in practice after updating the parameters of the model with an optimisation technique, we replace negative values with 0 and values greater than 1 with 1.

Rule Extraction and Injection

When the optimisation procedure terminates tuning the parameters, we can extract the rules encoded in the network, by inspecting the value of each parameter. Indeed, every weight of the model lies in the interval $[0, 1]$ and by the semantics given to their values, we can apply the argument in Lemma 3.4, to extract the literals included in the clauses of each layer HL^h, with $1 \le h \le l$ recursively. Algorithm 4 shows the general steps to decode the weights of the network until the last hidden layer HL^l into clauses (which represent the negated rule antecedents in $\mathcal{T}$). The algorithm treats every node of the network HL_j^h (Figure 3.1) as a clause. The j-th node of the input layer HL_j^0 is the clause v_j itself. Recursively, the j-th node of the hidden layer HL_j^h, $1 \le h \le l$ is the disjunction of clauses HL_i^{h-1} if $\boldsymbol{w}_{i,j}^h > z$, with a chosen threshold parameter $z \in [0, 1]$. The closer $\boldsymbol{w}_{i,j}^h$ is to 1, the more certain the clauses encoded by the node HL_i^{h-1} is a subclause of the clause encoded by the node HL_j^h. Algorithm 4 recursively builds clauses encoded by the

Algorithm 4: GET_CLAUSES

Input: a trained RIDDLE model $N : [0,1]^{2n} \rightarrow [0,1]^s$, a threshold z
Output: vector of s clauses encoded in N.
Let l be the number of hidden layers in N
$\mathsf{V}' \leftarrow (\mathsf{v}_1^0, \cdots, \mathsf{v}_{2n}^0)$
$\mathsf{V} \leftarrow \mathsf{V}'$
for $0 \leq h \leq l$ and $1 \leq j \leq \mathfrak{O}_h$ **do**
  add variable v_j^h to V
end for
$\mathcal{M} \leftarrow$ empty map
for $\mathsf{v} \in \mathsf{V}'$ **do**
  $\mathcal{M}(\mathsf{v}) \leftarrow \mathsf{v}$
end for
$h \leftarrow 1$
while $h \leq l$ **do**
  Let $\boldsymbol{w}^h$ be the weights of the layer HL^h
  for $1 \leq j \leq \mathfrak{O}_h$ **do**
    $\phi = \bot$
    for $1 \leq i \leq \mathfrak{I}_h$ **do**
      if $\boldsymbol{w}_{i,j}^h \geq z$ **then**
        $\phi \leftarrow \phi \vee \mathcal{M}\left(\mathsf{v}_i^{h-1}\right)$
      end if
      set $\mathcal{M}\left(\mathsf{v}_j^h\right) \leftarrow \phi$
    end for
  end for
  $h \leftarrow h + 1$
end while
return $(\mathcal{M}\left(\mathsf{v}_j^l\right) \mid 1 \leq j \leq \mathfrak{O}_l)$

Algorithm 5: GET_RULES

Input: a trained RIDDLE model $N : [0,1]^{2n} \rightarrow [0,1]^s$, a threshold z.
Output: rules encoded in N.
$\mathcal{H} \leftarrow \emptyset$
$\mathcal{C} \leftarrow$ GET_CLAUSES(N, z)
for each i-th element ϕ_i in $\mathcal{C}$ **do**
  for $1 \leq r \leq s$ **do**
    if $\phi_i \neq \bot$ and $\boldsymbol{\beta}_i^r < 1$ **then**
      add $(\neg\phi_i \rightarrow \mathsf{t}_r, 1 - \boldsymbol{\beta}_i^r)$ to $\mathcal{H}$
    end if
  end for
end for
return $\mathcal{H}$

input network, representing them with only variables that are associated to the nodes of the input layer. The clauses generated in the output correspond to the clauses ψ_i in Equation (3.1) or Equation (3.2).

To construct the final rules encoded in the network N, we run Algorithm 5. We recall that by Lemma 3.3 and Corollary 3.7, clauses encoded by nodes of the last hidden layer corresponds to the negated antecedent of a rule encoded in the network. Therefore, Algorithm 5 loops for each such clauses, and it forms a possibilistic rule. To compute the necessity value associated to the rule, we use the relation between necessity and possibility measures: $N(\phi) = 1 - \Pi(\neg\phi)$. In simple terms, Algorithm 5 exploits the direct correspondence between Equation (3.1) and Equation (3.4).

In our tests (next section), we observed that the parameters always collapse to either 0 or 1 after a sufficient number of parameter updates (epochs). The introduction of hidden layers can be considered a way of having predicate invention as in ILP settings [Muggleton et al., 2012]. Moreover, we can also manually inject rules of the form $(\phi \rightarrow \mathsf{t})$ to a RIDDLE instance before or after training. For doing that, due to Lemma 3.4, we just need to append to the operation LSE_{min} associated to target variable $\mathcal{T}$ in Equation (3.4), a layer $\mathsf{RL} : [0,1]^u \rightarrow [0,1]^v$ corresponding to the computation of injected and negated rule antecedents as described. In this way, a RIDDLE instance would look like in Equation (3.5):

$$\begin{aligned}\mathsf{RIDDLE}(\boldsymbol{x}) = (&\mathsf{LSE}^1_{min}(\mathsf{LSE}_{max}(\boldsymbol{\beta_i}^1, \mathsf{HL}^l_i(\boldsymbol{x}) \mid 1 \leq i \leq k) \\ &\qquad \mathsf{LSE}_{max}(\boldsymbol{\gamma_i}^1, \mathsf{RL}^1_i(\boldsymbol{x}) \mid 1 \leq i \leq c^1)) \\ &\cdots, \\ &\mathsf{LSE}^s_{min}(\mathsf{LSE}_{max}(\boldsymbol{\beta_i}^s, \mathsf{HL}^l_i(\boldsymbol{x}) \mid 1 \leq i \leq k) \\ &\qquad \mathsf{LSE}_{max}(\boldsymbol{\gamma_i}^s, \mathsf{RL}^s_i(\boldsymbol{x}) \mid 1 \leq i \leq c^s))),\end{aligned} \tag{3.5}$$

where for each target variable t_j, with $1 \leq j \leq s$, we have added c^j rules of the form $(\phi^j_i \rightarrow \mathsf{t}_j, 1-\gamma^j_i)$, for $1 \leq i \leq c^j$ to the rules encoded by the network. The nodes denoted with RL^j_i can be pictured as special nodes that take as input the values of the nodes in the first layer HL^0 such that the weight between node HL^0_m and RL^j_i is 1 if the clauses encoded by RL^j_i is supposed to include the variable v_m.

3.2 Experimental Results

We implemented the RIDDLE model in Python 3.9 that is fully integrated in the PyTorch [Paszke et al., 2019] ecosystem. The gradient of the model parameters are computed with PyTorch's automatic differentiation package and after the update, they are 'clamped' to the range $[0, 1]$ to preserve the correctness of the model. The implementation is available at the following link: `https://git.app.uib.no/Cosimo.Persia/riddle`. We conduct the experiments on an Ubuntu 18.04.5 LTS server with i9-7900X CPU at 3.30GHz, 32 physical cores, 8 GPUs NVIDIA A100 with 80GB, and 32GB RAM.

Test settings

Often, the features in the considered datasets include a mixture of nominal, continuous and integer fields. Using feature discretization, we divide continuous or integer values in 8 bins such that all bins for each feature have the same number of points. Each bin will be associated with a new variable that it is going to be set to 'true' if the value of the original value belongs to the respective bin. Missing values assign the value 'unknown' to all related new binarised variables. In this way, we can generate a set of interpretations $D \coloneqq \{\mathcal{I}_1, \ldots, \mathcal{I}_d\}$. Then, we can generate the dataset $\mathsf{D} \coloneqq \{\boldsymbol{x}^i \mid \mathcal{I}_i \in D\}$ as explained at the beginning of Section 3.1. D expresses the possibility values of variables and their negation for each interpretation in D. The first column in Table 3.1, shows the datasets that we considered for the benchmark. These are freely available at UCI machine learning repository [Dua and Graff, 2017], and commonly used to empirically test the performance of rule induction algorithms. Briefly:

- 'Anneal': In metallurgy, annealing is a heat treatment that alters the physical properties of a material. Examples are a description of states of an object before the heat treatment and the prediction is the final state of the object. The probability of an attribute to be missing is on average 60%.
- 'Audiology' : The input example is a description of ear characteristic of a patient. The target class denotes ear ailments. On average, 2% is probability of an attribute to be missing.
- 'Auto': Examples are descriptive information of an auto, associated with its value. That is whether it is expensive, medium, or low range. The probability of an attribute to be missing is on average 1%.
- 'Credit-A/Credit-G': Examples are credit card applications denoting information about a client. Each example is classified positively if a client has been approved

a credit card, negatively otherwise. 'Credit-A' concerns Australian clients, while 'Credit-G' only German clients. No missing attribute.

- 'Breast Cancer': Examples list features of a tumor, and it is classified either as benign or malignant. The probability of having a missing attributes is 1%.
- 'Chess': Each instances in this database is a sequence of 37 attribute values. Each instance is a board-descriptions for a chess endgame. The first 36 attributes describe the board. The last attribute is the classification whether the player will win or lose. No missing attributes.
- 'Glass': Examples are description of an object. Each of them is classified with the type of glass it belongs to: headlamp, window, etc. No missing attributes, but possibly uncertain measurements.
- 'Hepatitis': Instances are patients with relevant features and blood tests. They are classified as being able to survive or not. On average, 6% of attributes are missing.
- 'Horse': Examples describe the condition of sick horses. Each horse is associated with a class denoting whether the horse will survive the treatment, die or will be euthanised. The probability of having missing attributes is 15%.
- 'hypothyroid': Examples consist of patient observations and a binary class stating whether a person has hypothyroid or not. The probability of having a missing attributes is 5%.
- 'Lymphography': Instances denote properties of a tumor, with associated type: fibrosis, metastases, benign, etc. No missing attributes.
- 'Mushroom': The dataset is a samples corresponding to 23 species of gilled mushrooms. Each species is identified as definitely edible, definitely poisonous, or of unknown edibility and not recommended. The probability of having a missing attributes is 1%.
- 'Primary tumor': Instances are body measurements, and the associated class is the location of the tumor. The probability of having a missing attributes is 4%.
- 'Wine': Examples are chemical analysis of wines grown in Italy but derived from three different cultivars. They are classified according to the quantity of alcohol they generate after fermentation. There are no missing values, but it is possible to have imprecise measurements.

More details about the UCI datasets at `http://archive.ics.uci.edu/ml`. Most datasets have a substantial amount of missing values in selected features.

Evaluation

The RIDDLE model optimises a regression problem while training, so we use the MSE loss as a the measure to minimise. Compared with arbitrary linear/ReLU feedforward deep network architectures, RIDDLE performs slightly better (in addition to being explainable). Therefore, we will focus our comparison on the accuracy of the FURIA algorithm [Hühn and Hüllermeier, 2009], a prominent decision tree for fuzzy rule induction based on RIPPER [Cohen, 1995] that represents the state of the art algorithm in the field of propositional rule induction. We use the FURIA implementation freely available on Weka [Frank et al., 2005], and test it on classification tasks and compare based on the standard definition of the accuracy measure. To use the trained RIDDLE model for classification purposes, we look at RIDDLE output $(\Pi(\neg \mathsf{t}_1), \cdots, \Pi(\neg \mathsf{t}_s))$, and if $\Pi(\neg \mathsf{t}_i) \leq 0.4$, then the variable t_i is preferred over its negation $\neg \mathsf{t}_i$ and we assume that the variable t_i is predicted to be true. This is justified by the relation between possibility and necessity measures ($N(\mathsf{t}) = 1 - \Pi(\neg \mathsf{t})$). We carried our tests on the same benchmark datasets used by the aforementioned rule induction system.

Model selection

We split each dataset in 70%, 10%, and 20% for training, validation and test, respectively. Finding the best combinations of hyperparameters (number of layers, nodes per layers, learning rate etc.) can be a tedious task. But, we noticed that in general RIDDLE performs quite well already with small networks and extremely well with deeper configurations. We select the hyperparameters on a grid search fashion using Tune [Liaw et al., 2018]. The hyper-parameters are selected from a specified pool of values. The number of hidden layers varies from 1 to 10; the number of nodes per layer from 2 to 500; the batch size from 8 to 64; the learning rate from 0.1 to 0.001. We use SGD (Algorithm 3) to optimise the parameters.

The final models' sizes correlate positively with the number of variables given as input and with the number of rules that hold in the dataset. On average, the resulting network has 6 layers with 50 nodes. Each model is trained with a batch of size 16 over 100 epochs with a learning rate of 0.01, and with early-stopping. That is, we stop the training routine if the validation loss has not has not decreased by more than 0.01 for 10 subsequent epochs. Moreover, we fixed a weight-decay factor of 0.001. This means that the gradients used for updating the parameters are summed to the constant value 0.001.

					Accuracy	
					RIDDLE	FURIA
Dataset	**Inst.**	**Var.**	**Discrete Var.**	**MSE**		
Anneal	798	39	286	1e-4	97	97
Audiology	226	71	708	4e-3	**95**	91
Auto	205	26	354	3e-4	**98**	85
Credit-A	690	15	158	2e-1	87	**89**
Credit-G	1000	21	166	2e-2	**74**	72
Breast cancer	699	21	160	3e-2	**91**	90
Chess	3196	37	146	1e-2	91	**99**
Glass	214	10	72	4e-3	**94**	68
Hepatitis	155	20	122	4e-4	**86**	75
Horse	368	28	252	5e-3	83	**85**
Hypothyroid	3163	30	140	8e-3	95	95
Lymphography	148	19	118	9e-4	86	86
Mushroom	8124	19	234	1e-4	97	**98**
Primary tumor	339	18	60	6e-5	**94**	75
Wine	178	14	112	1e-4	**96**	90

Table 3.1: Accuracy of RIDDLE and FURIA compared with different datasets. The column 'Inst.' shows the number of instances, 'Var.' shows the number of variables in the original dataset and 'Discrete Var.' is the number variables after discretization and binarization.

				Model complexity			
				RIDDLE		FURIA	
Dataset	**Inst.**	**Var.**	**Discrete Var.**	**Size**	**Count**	**Size**	**Count**
Anneal	798	39	286	**2.9**	**17.1**	3.2	21.7
Audiology	226	71	708	**1.7**	**19.0**	2.1	19.3
Auto	205	26	354	2	3.4	2	**3.3**
Credit-A	690	15	158	**3.1**	**6.8**	4.1	7.3
Credit-G	1000	21	166	**2.2**	10.3	3.1	**10.2**
Breast cancer	699	21	160	**3.3**	**5.1**	3.6	9.9
Chess	3196	37	146	**4.6**	**15.3**	4.7	28.3
Glass	214	10	72	**1.8**	**5.9**	2.0	16.7
Hepatitis	155	20	122	3.1	3.4	**2.2**	4.1
Horse	368	28	252	**1.8**	5.9	2.7	**5.0**
Hypothyroid	3163	30	140	**4.8**	**11.3**	6.3	18.1
Lymphography	148	19	118	**2.1**	5.4	2.2	**5.3**
Mushroom	8124	19	234	2.8	**5.3**	**2.4**	6.0
Primary tumor	339	18	60	1.9	**3.6**	1.9	4.0
Wine	178	14	112	3.4	8.2	3.4	**8.1**

Table 3.2: Model complexity of RIDDLE and FURIA compared with different datasets. The column 'Inst.' shows the number of instances, 'Var.' shows the number of variables in the original dataset and 'Discrete Var.' is the number variables after discretization and binarization. 'Size' is the average number of literals per clause and 'Count' is the average number of clauses.

Dataset	Min	Max	Median	S.D.
Anneal	93	98	96	1.18
Audiology	94	96	95	0.63
Auto	95	99	98	2.48
Credit-A	66	92	86	4.12
Credit-G	68	82	76	5.65
Breast cancer	85	96	91	4.52
Chess	83	98	88	6.04
Glass	81	99	88	4.54
Hepatitis	71	89	86	6.83
Horse	78	86	84	5.71
Hypothyroid	91	98	94	4.37
Lymphography	83	88	87	2.91
Mushroom	94	98	97	1.17
Primary tumor	85	99	93	5.13
Wine	84	98	95	3.59

Table 3.3: Additional statistics from the empirical evaluation of RIDDLE obtained by running 40 training instances. The value 'min' and 'max' are the minimum and maximum accuracy found on test data. 'S.D.' is the computed standard deviation.

Results

Table 3.1 shows the results of our experiments concerning the MSE error and the accuracy on the test data compared with the FURIA decision tree model. Often, RIDDLE converges to a local minimum after only 30 epochs; and soon after the early-stopping routine stops the training. The average training time with the largest datasets (mushrooms, chess, hypothyroid), is 2 seconds for RIDDLE and 3 for FURIA. The 'Accuracy' columns in Table 3.2 shows that RIDDLE generalises often better from training data. FURIA performs better with complete information as in the 'Chess' dataset. But, even in such case RIDDLE outputs a simpler model. Concerning datasets with more missing data, such as glass or hepatitis, RIDDLE performs considerably better.

Also, Table 3.2 reports the size and the number of induced clauses found by each model per dataset. For instance, rules found by RIDDLE in the 'hepatitis' are

$$((1.9 \leq \mathsf{bilirubin}) \text{ and } \neg\mathsf{has\_ascites} \rightarrow \mathsf{dies}, 0.8),$$
$$((3.7 \leq \mathsf{albumin}) \text{ and } \mathsf{firm\_liver} \rightarrow \mathsf{lives}, 1).$$

Also on this aspect, RIDDLE shows a better model complexity with fewer and shorter rules, even when RIDDLE performed worse in terms of accuracy. This suggests a bias towards simpler models. As expected, when we manually inject to the RIDDLE model rules, that we know to hold in the given domain, with additional layers (end of Section 3.1), the performance improves.

Table 3.3 shows the statistical results after 40 different experiments. For each dataset, we stored the minimal and maximal accuracy. Then, we computed the median and standard deviation. From Table 3.3 we can conclude that RIDDLE has consistent performance in most of the datasets considered.

We remark that another advantage of RIDDLE is that the values provided with the rules have a clear meaning, in terms of necessity. Meanwhile, fuzzy approaches such as FURIA provide a weaker foundation for the interpretation of the values associated with the rules. Additionally, the necessity values also distinguish RIDDLE from approaches such as decision trees which, usually, do not provide a measure of a rule's reliability.

3.3 Discussion

We introduced RIDDLE; a novel deep learning architecture specialised in performing rule induction in the presence of missing, and uncertain data. RIDDLE is a white-box model architecture as its trained weights have a clear meaning concerning the decisions that the model takes while performing inference on the input. These weights can be translated into propositionally complete rules that are simpler than the rules found by established rule induction algorithms (Table 3.2). In addition, each rule is associated a certainty degree expressing the confidence of the model about the induced rule. Together with the capability of RIDDLE to incorporate background knowledge, and learn from incomplete or imprecise data, RIDDLE is a competitive algorithm for rule induction.

The proposed RIDDLE architecture is efficient and scales well with the size of the dataset as tested in our experiments. The matrix computation can be carried with general techniques, but we can additionally optimise it in out implementation, and speed-up both training and inference time. A better estimation of possibilities values associated to input variables can improve both the quality of the output rules and certainty degrees associated to them. For this reason, future work may focus on improving the data preprocessing. For example, we could investigate different methods of drawing possibilities distributions from imprecise data [Dubois and Prade, 1992, 2016]. Some are based on the connection between possibility distributions, statistics, uncertain probabilities, on qualitative analysis, and other interesting techniques. Another interesting project would be finding automated technique for discretising the original dataset. Indeed, the RIDDLE architecture requires the input to be discretised features. A technique for discretising and binarising continuous variables than minimises the total number of binary variables may be useful in domains with a high number of continuous measurement.

Chapter 4

Exact Learning of Possibilistic Logic Theories

The RIDDLE architecture, proposed in the previous chapter, is successful in delivering possibilistic rules induced from data. But other than the empirical evaluation, it is difficult to state properties or guarantees on the learning outcome. Possibilistic logic and its many applications [Dubois and Prade, 2014, 2015] have been extensively studied, but there are not many works that formally investigate the learnability of possibilistic theories. We partially cover this gap by studying whether possibilistic theories are learnable in Angluin's exact learning model [Angluin, 1988; Persia and Ozaki, 2020]. That is, under which conditions it is possible to guarantee exact identification of an unknown target concept. We show also whether (and under which conditions) it is possible to reduce polynomial time learnability results of classical logic formulas to polynomial time learnability of possibilistic logic formulas and vice-versa. A part of our contribution is the definition of a general and formal definition of learning problems that takes into account notions of the theory of computation Ozaki [2020b]. To the best of our knowledge, no books in machine learning provide a formal definition of learning algorithms in light of the theory of computation, in particular in the context of active learning. Watanabe (1990) addresses this need, but we give a more detailed and precise definition.

The exact learning model is an active model in which there is a learner that wants to learn an abstract target concept from data, that is obtained by the answers of queries asked to oracles. As defined in Section 2.4 of Chapter 2, a learner can be tasked to identify an unknown logic formula $\mathcal{T}$ based on the data acquired by checking if the current hypothesis is equivalent to $\mathcal{T}$ (Example 2.15). The oracle, also called the teacher, is assumed to know the vocabulary and the target concept, and whenever it is asked a query by the learner, it is expected to answer truthfully. The most studied communication protocol

in this model allows the learner to ask membership and equivalence queries. We also consider other types of queries in Angluin's model [Angluin, 1988], such as *superset* and *subset* queries. When studying the learnability or reducibility of learning problems, we consider cases in which only membership, equivalence, superset, subset, and both membership and equivalence or both superset and subset queries can be posed by the learner while identifying the target concept.

Concerning learnability results, we show that when the number of digits used to express necessity degrees, also called *precision*, occurring in the target is not known, we cannot (exactly) learn possibilistic theories with only membership, superset, or subset queries. When the learner can ask or simulate equivalence queries or when the precision of the target is known, we have positive learnability results (Section 4.2). We also get positive results because these queries together can simulate both membership and equivalence queries. When the learner does not know the precision of the necessity degrees of the target and it can ask membership and equivalence queries, for a large class of problems, polynomial time learnability can be transferred from classical logic to the respective possibilistic extension. If the precision of the target is known by the learner, it is possible to transfer learnability results into the possibilistic settings with other queries. For instance, if we allow the learner to ask only membership queries, and we assume that the maximal precision of valuations in the target is fixed and known by the learner, then polynomial time learnability of a classical logic can also be transferred to the respective possibilistic extension.

As a consequence of our mentioned results, we establish, for instance that since propositional Horn [Angluin et al., 1992; Frazier and Pitt, 1993a] and fragments of first-order Horn [Arimura, 1997b; Konev et al., 2018; Reddy and Tadepalli, 1998b] are exactly learnable in polynomial time (with both kinds of queries), their respective possibilistic extensions are also learnable in polynomial time. Table 4.1 summarises our results. The first column shows the results concerning learnability. ✓ means a positive result, p indicates that is assumed that the learner knows the maximal precision of valuations present in the target. ✗ and $\not p$ indicate their respective negation. The third column shows the transferability of polynomial time learnability results from classical settings $\mathfrak{F}$ to possibilistic settings $\mathfrak{F}_\pi$. The fourth column shows the result in the other direction. $\checkmark^*$ and $\checkmark^\dagger$ indicate respectively, that the learnability result holds only when its classical setting is learnable and when there is a positive bounded learner for that learning framework, that is when the learner guarantees that its hypothesis is a logical consequence of the target at every step. As polynomial time learnability in the exact model is transferable to the classical probably approximately correct (PAC) [Valiant, 1984] model extended with membership queries, our work also establishes such results in this model.

Queries	Learnability		Pol. Time. Reduction $\mathfrak{F} \to \mathfrak{F}_\pi$		$\mathfrak{F}_\pi \to \mathfrak{F}$	
	p	$\not{p}$	p	$\not{p}$	p	$\not{p}$
MQ	✓*	✗	✓	✗	✓	–
SbQ	✓*	✗	✓	✗	✓	–
SpQ	✓*	✗	✓	✗	✓	–
EQ	✓	✓	✓†	✓†	✓	✓
MQ,EQ	✓	✓	✓	✓	✓	✓
SbQ,SpQ	✓	✓	✓	✓	✓	✓

Table 4.1: Each row summarises the outcome when a learner can ask only membership (MQ), subset (SbQ), superset (SpQ), equivalence (EQ) or both (MQ,EQ), (SbQ,SpQ).

In Section 4.1, we introduce the novel and necessary notions of computational learning theory. In Section 4.2, we investigate whether possibilistic logic theories can be learned and, in Section 4.3, we show transferability of polynomial time learnability results. We conclude with a discussion and directions for future work. For conciseness, we defer some proofs to Appendix A.2 as they employ the same strategy of other proofs in this chapter.

4.1 Learning System

In order to study learning problems taking into account the theory of computation, we define the learner as a deterministic multitape Turing machine (DMTM) and the teacher as a non-deterministic multitape Turing machine (NMTM) [Sipser, 1997]. A DMTM with k tapes is a tuple $M = (Q, \Sigma, \Theta, q_0, q_f)$ where Q is a finite set of states, Σ is a finite alphabet including the blank symbol, $\Theta : (Q \setminus \{q_f\}) \times \Sigma^k \to Q \times \Sigma^k \times \{l, r\}^k$ is the transition function, q_0 is the initial state where the computation starts and q_f is the final state where the computation ends. The expression $\Theta(q, s_1, \cdots, s_k) = (q', s'_1, \cdots, s'_k, D_1, \cdots, D_k)$ means that if the DMTM is in state q and the head of the tape $1 \leq i \leq k$ reads the symbol s_i, then M transitions to state q', each head in tape i moves to the direction D_i and the symbol s'_i is written to every tape i at the position the head points to. A NMTM is similar to a DMTM with the only difference that Θ is a relation, i.e. for every $q \in Q$, $s_i \in \Sigma$ we have $\Theta(q, s_1, \cdots, s_k) \subseteq 2^{Q \times \Sigma^k \times \{l,r\}^k}$. The learner has special states called *query state*, for each type of query it is allowed to ask, and the teacher has *answer states*, one for each type query it supports. A *configuration* of a DMTM or NMTM (MTM) with k tapes is a k-tuple $(w'_1 q w_1, \cdots, w'_k q w_k)$ where $w'_i, w_i \in \Sigma^*$ and $q \in Q$. A configuration captures a snapshot of the computation of a MTM, for every $1 \leq i \leq k$, $w'_i q w_i$ means that the MTM is on state q, on tape i it is written the word $w'_i w_i$ and its head points to the first symbol in w_i. A computation of a MTM is a sequence of configurations $((w'_1 q_0 w_1, \cdots, w'_k q_0 w_k)_1, \cdots (u'_1 q u_1, \cdots, u'_k q u_k)_n, \cdots)$ such

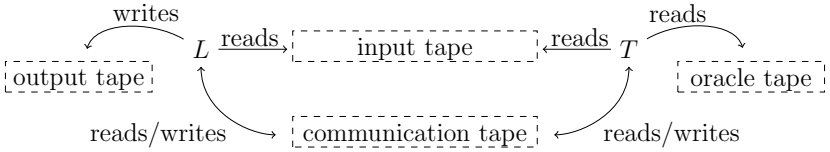

Figure 4.1: Graphical representation of a learning system.

that it starts from the initial state q_0 and each pair of consecutive configurations satisfies the constraints imposed by Θ. For example, for two consecutive configurations $(w'_1 q s_1 w_1, \cdots, w'_k q s_k w_k)_i, (u'_1 r s'_1 u_1, \cdots, u'_k r s'_k u_k)_{i+1}$ in a computation of DTM, we have $\Theta(q, s_1, \cdots, s_k) = (r, s'_1, \cdots, s'_k, D_1, \cdots, D_k)$.

Concerning the formal description of the learning protocol, we are going to use the formal computational model called 'learning system' [Ozaki, 2020b; Watanabe, 1990]. A *learning system* is a pair (L, T) of multitape Turing machines where L is the learner and T is the teacher. In a learning system there are four types of tapes:

- a read-only *input tape*, shared by L and T;
- a read-write *communication-tape* shared by L and T;
- a read-only *oracle tape* accessed only by T;
- a write-only *output tape* accessed only by L.

L is a DMTM with three tapes: input, communication and output tape. T is NMTM with input, communication and oracle tape (Figure 4.1). The computation of the learning system (L, T) executes L and T, in turn, starting from L. L starts its computation from its initial state and it stops (it does not halt) in *query state* (associated the respective query) upon writing a query string on the communication tape. T starts from its initial state, reads the query and it writes the answer in the communication tape, it stops in the *answer state* corresponding to the type of query answered, and L resumes its execution. L and T alternate their execution until L enters its final state. When this happens, the learning system halts.

A *computation of a learning system* (L, T) *on an input word* w is a tree $\mathsf{T}_{L,T,w}$ whose paths are sequences of successive configurations determined by the transition relations of L and T. The root is the configuration corresponding to the initial state of L. The existence of branches are due to the non-determinism of T. If a path p (rooted) in $\mathsf{T}_{L,T,w}$ contains a configuration corresponding to the final state of L (which is also the last configuration of every path, if present) we say that p is *terminated* or *a terminated path.*

Otherwise, we say that p is *unterminated* or *an infinite path*. The length of a path p is the number of configurations in it but configurations corresponding to the computation of T in p are not part of the count. We additionally write p^i with $i \in \mathbb{N}^+$ to denote the prefix of the path p that from the initial configuration of L it includes every subsequent configuration in p until the i-th configuration. The *output of a terminated path* p is the content of the output tape in the last configuration of p.

In this model, for a learner, posing a query to an oracle means writing down the query in an (additional) communication tape, entering in a query state, and waiting. The oracle then writes the answer in the communication tape, enters in an answer state, and stops. After that, the learner resumes its execution and can now read the answer in the communication tape. The proposed model of computation can be generalised to cases of multiple learners (DMTM) and teachers (NMTM).

Let $\mathfrak{F} = (\mathcal{E}, \mathcal{L}, \mu)$ be a learning framework. When we write $(L_{\mathfrak{F}}, T_{\mathfrak{F}})$ we mean the learning system (L, T) where an arbitrary target $\mathcal{K} \in \mathcal{L}$ is written in the oracle tape, strings written in the output tape are always in $\mathcal{L}$, and queries posed by the learner written in the communication tape are always in $\mathcal{E}$. If a specific $\mathcal{K} \in \mathcal{L}$ is written in the oracle tape, we write $T_{\mathfrak{F}}(\mathcal{K})$. If it is clear from the context, we may omit the subscript $\mathfrak{F}$ from $(L_{\mathfrak{F}}, T_{\mathfrak{F}})$. A teacher T is said to be *terminating* if for every $\mathcal{K} \in \mathcal{L}$, $T(\mathcal{K})$ it is guaranteed to terminate for every possible query it can answer. In this work, we are going to consider learning frameworks where we assume that a terminating teacher always exists.

Remark 4.1. *Computing an answer to a query can be an undecidable task. In particular, in this work, membership queries are entailment queries and entailment in e.g. FO-logic is well known to be undecidable. If the entailment problem is decidable in a fragment of FO-logic, that is, if there is a terminating teacher that can answer membership queries, then the same problem is decidable in the possibilistic extension of that logic [Lang, 2000].*

Given a learning framework $\mathfrak{F} = (\mathcal{E}, \mathcal{L}, \mu)$ and the learning system $(L_{\mathfrak{F}}, T_{\mathfrak{F}}(\mathcal{K}))$ where $\mathcal{K} \in \mathcal{L}$ is the target, and $T_{\mathfrak{F}}(\mathcal{K})$ is terminating, we say that:

- $L_{\mathfrak{F}}$ *calls the oracle* $\mathsf{MQ}_{\mathfrak{F},\mathcal{K}}$ *with input* $e \in \mathcal{E}$, whenever $L_{\mathfrak{F}}$ writes e in the communication tape, enters in the membership query state and stops (it does not halt). $T_{\mathfrak{F}}$ resumes its execution, it writes the answer 'yes' in the communication tape if $e \in \mu(\mathcal{K})$, 'no' otherwise, it stops in membership answer state, and $L_{\mathfrak{F}}$ resumes its execution. Let p be a path in $\mathsf{T}_{L_{\mathfrak{F}}, T_{\mathfrak{F}}(\mathcal{K}), \Sigma_{\mathcal{K}}}$. We say 'the i-th call to $\mathsf{MQ}_{\mathfrak{F},\mathcal{K}}$ with input $e \in \mathcal{E}$ in p' to refer to the i-th configuration in p where $L_{\mathfrak{F}}$ calls $\mathsf{MQ}_{\mathfrak{F},\mathcal{K}}$ with input the example e.

- $L_{\mathfrak{F}}$ *calls the oracle* $\mathsf{EQ}_{\mathfrak{F},\mathcal{K}}$ *with input* $\mathcal{H} \in \mathcal{L}$, whenever $L_{\mathfrak{F}}$ writes $\mathcal{H}$ in the commu-

nication tape, enters in equivalence query state and it stops (it does not halt). $T_{\mathfrak{F}}$ resumes its execution, it writes the answer $e \in \mu(\mathcal{K}) \oplus \mu(\mathcal{H})$ in the communication tape if $\mu(\mathcal{K}) \neq \mu(\mathcal{H})$, 'yes' otherwise, it stops in equivalence answer state, and $L_{\mathfrak{F}}$ resumes its execution. Analogously as before, we say 'the i-th call to $\mathsf{EQ}_{\mathfrak{F},\mathcal{K}}$ with input $\mathcal{H}$ in p' to refer to the i-th configuration in p where $L_{\mathfrak{F}}$ calls $\mathsf{EQ}_{\mathfrak{F},\mathcal{K}}$ with input $\mathcal{H}$. We say the same for calls to the superset $\mathsf{SbQ}_{\mathfrak{F},\mathcal{K}}$ or subset $\mathsf{SpQ}_{\mathfrak{F},\mathcal{K}}$ oracles but the counterexample returned is $e \in \mu(\mathcal{H}) \setminus \mu(\mathcal{K})$ in the case of $\mathsf{SbQ}_{\mathfrak{F},\mathcal{K}}$ and $e \in \mu(\mathcal{K}) \setminus \mu(\mathcal{H})$ in the case of $\mathsf{SpQ}_{\mathfrak{F},\mathcal{K}}$.

Given a learning system $(L, T(\mathcal{K}))$ on input X, we refer to it saying 'L *attempts to learn* $\mathcal{K}$ *on input* X'. We write $Y \in (L, T(\mathcal{K}))(X)$ if there is a terminated path in $\mathsf{T}_{L,T(\mathcal{K}),X}$ where the output is Y.

We are now ready to define notions of learnability for learning problems. We say that $\mathfrak{F} = (\mathcal{E}, \mathcal{L}, \mu)$ is (exactly) *learnable* if there is a a terminating T, and there is a learner L such that for any $\mathcal{K} \in \mathcal{L}$, the computation tree $\mathsf{T}_{L,T'(\mathcal{K}),\Sigma_{\mathcal{K}}}$ is finite (every path rooted in $\mathsf{T}_{L,T'(\mathcal{K}),\Sigma_{\mathcal{K}}}$ is a terminated run) and every $\mathcal{H} \in (L, T'(\mathcal{K}))(\Sigma_{\mathcal{K}})$ satisfies $\mu(\mathcal{H}) = \mu(\mathcal{K})$. In this case L is said to be *a learner for* $\mathfrak{F}$ (Remark 4.2).

Remark 4.2. We can assume that the signature Σ given as input to the learning system $(L, T'(\mathcal{K}))(\Sigma)$ can contain more symbols, that is $\Sigma \supseteq \Sigma_{\mathcal{K}}$, and it does not affect the computation of the learning system.

If additionally, in every path rooted in $\mathsf{T}_{L,T'(\mathcal{K}),\Sigma_{\mathcal{K}}}$ the sum of the number of configurations corresponding to the computation of L is bounded by a polynomial w.r.t. $|\mathcal{K}|$ and the size of the largest counterexample returned by the teacher so far (if equivalence, superset, or subset queries are allowed), $\mathfrak{F}$ is *polynomial time learnable* and L is said to be a *polynomial time learner for* $\mathfrak{F}$.

If a learner L is such that at all times its built hypothesis $\mathcal{H}$ satisfies $\mu(\mathcal{H}) \subseteq \mu(\mathcal{K})$, then L is said to be a *positive bounded learner*.

Remark 4.3. *Let $\mathfrak{F} = (\mathcal{E}, \mathcal{L}, \mu)$ be an FO learning framework and let $\mathcal{K} \in \mathcal{L}$ be the target and let $\mathcal{H} \in \mathcal{L}$ be the hypothesis constructed by a learner. If the learner has access to $\mathsf{MQ}_{\mathfrak{F},\mathcal{K}}$ then we can assume w.l.o.g. that if there is a learner for $\mathfrak{F}$ then there is positive bounded learner for $\mathfrak{F}$: the learner can check whether each $\phi \in \mathcal{H}$ is entailed by $\mathcal{K}$. The same holds for $\mathfrak{F}_\pi$.*

As similarly introduced by Ozaki (2020b), we denote by EL(MQ,EQ) and ELP(MQ,EQ) the classes of all learning frameworks that are, respectively, exactly learnable and exactly

learnable in polynomial time with membership and equivalence queries. EL(SbQ,SpQ), and ELP(SbQ,SpQ) denote similarly the case when subset and superset queries can be asked. Analogously, EL(MQ), ELP(MQ), EL(EQ), ELP(EQ), EL(SbQ), ELP(SbQ) EL(SpQ), ELP(SpQ) are the respective classes of learning frameworks where only one type of query can be asked. In this work, polynomial time learnability assumes tractable complexity of some reasoning tasks (Remark 4.4).

Remark 4.4. *Often while proving polynomial time learnability results, we ask the learner to check if a formula is entailed by a classical or possibilistic KB. This means that those results hold only for logics which complexity of entailment-check is tractable. In this work, we assume that if a learning framework is polynomial time learnable (in any learning model), the logic used to define the hypothesis space allows to check if an example e is entailed by a hypothesis $\mathcal{H}$ in polynomial time w.r.t. $|\mathcal{H}|$ and $|e|$.*

Let $\mathfrak{F} = (\mathcal{E}, \mathcal{L}, \mu)$ be a learning framework. By possibilistic extension of $\mathcal{H} \in \mathcal{L}$ we mean the set (with infinite size) of possibilistic KBs of the form $\{(\phi, \alpha) \mid \phi \in \mathcal{H}, \alpha \in (0, 1]\}$. The possibilistic extension $\mathfrak{F}_\pi$ of $\mathfrak{F}$ is the triple $(\mathcal{E}_\pi, \mathcal{L}_\pi, \mu_\pi)$ where $\mathcal{L}_\pi$ is the set of all elements in the possibilistic extensions of each $\mathcal{H} \in \mathcal{L}$, $\mathcal{E}_\pi$ is the set of all possibilistic formulas entailed by an element of $\mathcal{L}_\pi$, and μ_π is the entailment relation (Example 4.5).

Example 4.5. The possibilistic extension of the learning framework $\mathfrak{F}$ of Example 2.10 in Chapter 2 is $\mathfrak{F}_\pi = (\mathcal{E}_\pi, \mathcal{L}_\pi, \mu_\pi)$ where $\mathcal{L}_\pi$ is the set of all possibilistic propositional KBs on the set of variables $\mathsf{V} = \{\mathsf{v}_1, \mathsf{v}_2\}$ and $\mathcal{E}_\pi$ is the set of all possibilistic formulas expressible with variables V. An element of $\mathcal{L}_\pi$ may be $\mathcal{H}_\pi = \{(\neg \mathsf{v}_1 \vee \mathsf{v}_2, 0.3), (\mathsf{v}_1, 0.8)\}$ and for instance the example $(\mathsf{v}_2, 0.3) \in \mathcal{E}_\pi$ satisfies $(\mathsf{v}_2, 0.3) \in \mu_{\pi_\mathcal{H}}$. ◁

In our work, we consider the problem of learning targets represented in decidable fragments of FO logic or in their possibilistic extensions. For this reason, when defining learning frameworks $(\mathcal{E}, \mathcal{L}, \mu)$ where $\mathcal{L}$ and $\mathcal{E}$ are formulas, we assume that μ is the entailment relation. That is, for every $\mathcal{H} \in \mathcal{L}$, $\mu(\mathcal{H}) = \{\phi \in \mathcal{E} \mid \mathcal{H} \models \phi\}$ (Example 2.10 or Example 4.5). We call such an $\mathfrak{F}$ an *FO learning framework*. We say that $\mathfrak{F}$ is *falsifiable* if $\mathcal{L}$ contains a falsifiable FO KB, *non-trivial* if $\mathcal{L}$ contains a non-trivial FO KB; and that it is *safe* if for all $\mathcal{H} \in \mathcal{L}$, then $\mathcal{H}' \subseteq \mathcal{H}$ implies that $\mathcal{H}' \in \mathcal{L}$.

Remark 4.6. *Polynomial time learnability results of learning frameworks do not depend on the complexity of reasoning in the logic used to express hypotheses or examples. For example, 3-CNF formulas are exactly learnable in polynomial time (with only equivalence queries) [Angluin, 1988] while entailment in 3-CNF is NP-hard. On the other hand, the logic $\mathcal{EL}$ is not learnable from entailments in polynomial time with membership and equivalence queries even though the complexity of the entailment problem in this logic is in* PTIME *[Konev et al., 2016, 2018].*

4.2 Learnability

In this section, we investigate the learnability of possibilistic logic KBs in the learning from entailment setting and in cases where the learner can ask only membership, only superset, only subset, or only equivalence queries. That is, we want to identify queries that allow for an identification of the target without taking into account efficiency or resources limitations.

We start with some important properties of possibilistic logic that are going to be used during the proofs in the next sections. Some points can be easily found in the literature, The proof of Point 5 is presented in the literature [Persia and Ozaki, 2020] and Points 3 to the best of our knowledge, is not explicitly proven in any previous work.

Proposition 4.7. *Let $\mathcal{H}$ be a possibilistic KB. For every possibilistic formula (ϕ, α),*

1. $\mathsf{inc}(\mathcal{H}) = 0$ *iff* $\mathcal{H}^*$ *is consistent;*
2. $\mathcal{H} \models (\phi, \alpha)$ *iff* $\mathcal{H}_\alpha \models (\phi, \alpha)$*;*
3. $\mathcal{H} \models (\phi, \alpha)$ *iff* $\mathcal{H}^*_\alpha \models \phi$*;*
4. $\mathcal{H} \models (\phi, \alpha)$ *iff* $\alpha \leq \mathsf{val}(\phi, \mathcal{H})$*; and*
5. $\mathcal{H} \models (\phi, \alpha)$ *implies* $\mathsf{val}(\phi, \mathcal{H}) \in \mathcal{H}^v \cup \{1\}$.

Proof.

1. Point 1 of Proposition 3.5.2 in Dubois and Prade (2014).
2. Proposition 3.5.6 in Dubois and Prade (2014).
3. By Point 2 of Proposition 4.7, we have that $\mathcal{H} \models (\phi, \alpha)$ iff $\mathcal{H}_\alpha \models (\phi, \alpha)$. By Proposition 2.9, it holds that $\mathcal{H}_\alpha \models (\phi, \alpha)$ iff $\mathcal{H}_\alpha \cup \{\neg\phi, 1\} \models (\bot, \alpha)$ or equivalently $\mathsf{val}(\phi, \mathcal{H}_\alpha) = \mathsf{inc}(\mathcal{H}_\alpha \cup \{\neg\phi, 1\}) > 0$ and by definition of valuation, we know that $\alpha > 0$. By Point 1 of Proposition 4.7, it follows that $\mathsf{inc}(\mathcal{H}_\alpha \cup \{\neg\phi, 1\}) > 0$ iff $(\mathcal{H}_\alpha \cup \{\neg\phi, 1\})^*$ is inconsistent iff $\mathcal{H}^*_\alpha \cup \{\neg\phi\}$ is inconsistent. It classically follows that $\mathcal{H}^*_\alpha \cup \{\neg\phi\}$ is inconsistent iff $\mathcal{H}^*_\alpha \models \phi$. At this point we have proven that $\mathcal{H} \models (\phi, \alpha)$ iff $\mathcal{H}^*_\alpha \models \phi$.
4. Property 1 at page 453 in Dubois et al. (1994).

5. Let Ω be the set of interpretations of the language used to represent the KB $\mathcal{H}$. By definition of $\pi_{\mathcal{H}}$, for all $\mathcal{I} \in \Omega$, $\pi_{\mathcal{H}}(\mathcal{I})$ is either 1 or $1-\beta$ for some $\beta \in \mathcal{H}^v$. Let $N_{\mathcal{H}}$ be the necessity measure induced by $\pi_{\mathcal{H}}$. By definition of $N_{\mathcal{H}}$, we have $N_{\mathcal{H}}(\phi) = \inf\{1-\pi_{\mathcal{H}}(\mathcal{I}) \mid \mathcal{I} \in \Omega,\ \mathcal{I} \models \neg\varphi\}$. Then, $N_{\mathcal{H}}(\phi) \in \mathcal{H}^v \cup \{0,1\}$ (recall that $\inf\{\}$ is 1, which is the case for tautologies). By the semantics of possibilistic logic, $N_{\mathcal{H}}(\phi) = \mathsf{val}(\phi, \mathcal{H})$ [Dubois et al., 1994, Corollary 3.2.3]. As (ϕ, α) is a possibilistic formula, $\alpha > 0$. So, by Point 2, $N_{\mathcal{H}}(\phi) = \mathsf{val}(\phi, \mathcal{H}) \in \mathcal{H}^v \cup \{1\}$. □

Let $\alpha \in (0,1]$ be a valuation. We denote by $\downarrow_p(\alpha)$ the operator that checks if two numbers are equal up to precision p (it truncates the number after precision p). For example $\downarrow_3(0.12345) = 0.123 = \downarrow_3(0.12377)$ but $\downarrow_3(0.124) \neq 0.12345$. We denote by $\uparrow_p(\alpha)$ the number α truncated at precision p and if the truncated part is greater than 0, then the last digit (the digit at the p-th position after the comma) is increased by one. For example, if $p = 2$ then $\uparrow_p(0.12001) = 0.13$, $\uparrow_p(0.3200) = 0.32$ and $\uparrow_p(0.3) = 0.30$. We write $\mathsf{prec}(\alpha)$ for the precision of the number (or alternatively valuation) α and $\mathsf{prec}(\mathcal{T}) = \sup\{\mathsf{prec}(\alpha) \mid (\phi, \alpha) \in \mathcal{T}\}$ the precision of the KB $\mathcal{T}$. Given an interval I, we write I_p for the set containing all $\alpha \in I$ that satisfy $\mathsf{prec}(\alpha) = p$. Recall that we write $\mathbb{N}^+$ for the set of positive natural numbers. Given $p \in \mathbb{N}^+$, we denote by $\mathfrak{F}^p_\pi = (\mathcal{E}_\pi, \mathcal{L}^p_\pi, \mu_\pi)$ the p-possibilistic extension of $\mathfrak{F}$, which results from removing every $\mathcal{H} \in \mathcal{L}_\pi$ in $\mathfrak{F}_\pi$ that does not satisfy $\mathsf{prec}(\mathcal{H}) = p$.

We start the study of whether any possibilistic learning framework $\mathfrak{F}_\pi$ is learnable with only membership queries. The main difficulty in learning with only membership queries (even for plain FO settings) is that the learner would 'not know' whether it has found a formula equivalent to a (non-trivial) target. Example 2.14 in Chapter 2 provides the description of a learning framework non-learnable with only membership queries. For possibilistic theories, another difficulty arises even for the propositional case. As the precision of a formula can be arbitrarily high, the learner may not know when to stop (e.g., is the target $\{(\mathsf{v}, 0.1)\}$? or $\{(\mathsf{v}, 0.11)\}$? If it is not $\{(\mathsf{v}, 0.11)\}$, is it $\{(\mathsf{v}, 0.101)\}$?). Theorem 4.8 states that, indeed, except for trivial cases, learnability cannot be guaranteed.

Theorem 4.8. *Let $\mathfrak{F}$ be a non-trivial FO learning framework. $\mathfrak{F}_\pi$ is not in* $\textsc{El}(\mathsf{MQ})$.

Sketch. Proof in Appendix A.2. Any learner L for the possibilistic extension $\mathfrak{F}_\pi$ will not be able to determine if the precision of a formula ϕ entailed by the target is higher than the one estimated by the learner with a finite amount of membership queries. After every membership query the learner is left with an infinite set of candidate hypotheses. □

If the precision of the target is known or fixed, learnability of an FO learning framework can be transferred to its possibilistic extension. We state this in Theorem 4.11. To show

it, we use the following technical results that are also going to be used in some other proofs in the next sections.

Lemma 4.9. *Let $\mathcal{T}$ be a possibilistic KB. Let I be a finite set of valuations such that $\mathcal{T}^v \subseteq I$. If for each $\alpha \in I$ there is some FO KB $\mathcal{K}^*_\alpha$ such that $\mathcal{K}^*_\alpha \equiv \mathcal{T}^*_\alpha$ then, it holds that $\mathcal{T} \equiv \{(\phi, \alpha) \mid \phi \in \mathcal{K}^*_\alpha, \alpha \in I\}$.*

Proof. Let $\mathcal{H} = \{(\phi, \alpha) \mid \phi \in \mathcal{K}^*_\alpha \equiv \mathcal{T}^*_\alpha, \alpha \in I\}$. Assume $\mathcal{H} \models (\phi, \gamma)$. If $\gamma = 1$ and $\gamma \notin I$ then ϕ is a tautology. In this case, for all $\beta \in (0, 1]$, $\mathcal{T} \models (\phi, \beta)$. Suppose this is not the case. By Points 4 and 5 of Proposition 4.7, $\gamma \leq \alpha$, where $\alpha = \mathsf{val}(\phi, \mathcal{H}) \in \mathcal{H}^v \cup \{1\}$. Also, $\mathcal{H} \models (\phi, \alpha)$. By construction of $\mathcal{H}$, $\mathcal{H}^v = I$, so $\alpha \in I$. Moreover, for every $\beta \in I$, we know that $\mathcal{H}^*_\beta = \mathcal{K}^*_\beta$. Therefore $\mathcal{K}^*_\alpha \equiv \mathcal{H}^*_\alpha$. By Point 3 of Proposition 4.7, $\mathcal{H} \models (\phi, \alpha)$ implies $\mathcal{H}^*_\alpha \models \phi$. Then, $\mathcal{K}^*_\alpha \models \phi$. As $\mathcal{K}^*_\alpha \equiv \mathcal{T}^*_\alpha$, we have that $\mathcal{T}^*_\alpha \models \phi$. Again by Point 3 of Proposition 4.7, $\mathcal{T}^*_\alpha \models \phi$ iff $\mathcal{T} \models (\phi, \alpha)$. Since $\alpha \geq \gamma$, $\mathcal{T} \models (\phi, \gamma)$ by Point 2. The other direction can be proven similarly. □

Lemma 4.10. *Let $\mathcal{H}$ be an FO KB and let $\mathcal{T}$ be the possibilistic KB $\{(\phi, 1) \mid \phi \in \mathcal{H}\}$. For every possibilistic formula (ϕ, α), we have that $\mathcal{H} \models \phi$ iff $\mathcal{T} \models (\phi, \alpha)$.*

Proof. If $\mathcal{T} \models (\phi, \alpha)$, since $\mathcal{T}^* \models \mathcal{T}^*_\alpha$ and $\mathcal{H} = \mathcal{T}^*$, $\mathcal{H} \models \phi$. If $\mathcal{H} \models \phi$, by construction $\mathcal{T}^*_1 \models \phi$. By Point 3 of Proposition 4.7, $\mathcal{T}^*_1 \models \phi$ iff $\mathcal{T} \models (\phi, 1)$, so, for all $\alpha \in (0, 1]$, $\mathcal{T} \models (\phi, \alpha)$. □

Theorem 4.11. *Let $\mathfrak{F} = (\mathcal{E}, \mathcal{L}, \mu)$ be a safe FO learning framework. For all $p \in \mathbb{N}^+$, let $\mathfrak{F}^p_\pi = (\mathcal{E}_\pi, \mathcal{L}^p_\pi, \mu_\pi)$ be its p-possibilistic extension. Then, $\mathfrak{F} \in \textsc{El}(\mathsf{MQ})$ iff $\mathfrak{F}^p_\pi \in \textsc{El}(\mathsf{MQ})$.*

Proof. ($\Rightarrow$) We show first that if $\mathfrak{F}$ is learnable, then $\mathfrak{F}^p_\pi$ is learnable. Let $\mathcal{T} \in \mathcal{L}^p_\pi$ be the target, and let $T_{\mathfrak{F}_\pi}$ be a terminating teacher (Remark 4.1). We describe the action of a learner $L_{\mathfrak{F}_\pi}$ such that the computation tree of $(L_{\mathfrak{F}_\pi}, T_{\mathfrak{F}_\pi}(\mathcal{T}))$ with input $\Sigma_\mathcal{T}$ has finite depth and $\mathcal{H} \in (L_{\mathfrak{F}_\pi}, T_{\mathfrak{F}_\pi}(\mathcal{T}))(\Sigma_\mathcal{T})$ satisfies $\mathcal{H} \equiv \mathcal{T}$. The fact that $\mathfrak{F}$ is in $\textsc{El}(\mathsf{MQ})$ implies the existence of a finite $\mathsf{T}_{L,T(\mathcal{K}),\Sigma_\mathcal{K}}$ for every $\mathcal{K} \in \mathcal{L}$ where L is a learner for $\mathfrak{F}$, T is a terminating teacher and, if $\mathcal{H} \in (L, T(\mathcal{K}))(\Sigma_\mathcal{K})$, it satisfies $\mathcal{H} \equiv \mathcal{K}$. For every $\alpha \in (0, 1]_p$, $\mathcal{T}^*_\alpha \in \mathcal{L}$ because $\mathfrak{F}$ is safe.

Description of $L_{\mathfrak{F}_\pi}$'s steps. For each $\alpha \in (0, 1]_p$, the learner $L_{\mathfrak{F}_\pi}$ repeats the same steps performed by the learner L in an arbitrary path p in $\mathsf{T}_{L,T(\mathcal{T}^*_\alpha),\Sigma_{\mathcal{T}^*}}$. At the i-th call to $\mathsf{MQ}_{\mathfrak{F},\mathcal{T}^*_\alpha}$ with input ϕ in p, $L_{\mathfrak{F}_\pi}$ calls $\mathsf{MQ}_{\mathfrak{F}_\pi,\mathcal{T}}$ with (ϕ, α) as input. This is done in finitely many steps. By Point 3 of Proposition 4.7, $\mathsf{MQ}_{\mathfrak{F},\mathcal{T}^*_\alpha}(\phi) = \mathsf{MQ}_{\mathfrak{F}_\pi,\mathcal{T}}(\phi, \alpha)$, therefore $L_{\mathfrak{F}_\pi}$ is able to perform every step made by L in p. Since $\mathsf{T}_{L,T(\mathcal{K}),\Sigma_\mathcal{K}}$ is finite and $(0, 1]_p$ contains finitely many valuations, $L_{\mathfrak{F}_\pi}$ will be able to find a $\mathcal{K}^*_\alpha \equiv \mathcal{T}^*_\alpha$ for each $\alpha \in (0, 1]_p$ in finitely many steps. By Lemma 4.9, $\mathcal{T} \equiv \mathcal{H} = \{(\phi, \alpha) \mid \phi \in \mathcal{K}^*_\alpha, \alpha \in (0, 1]_p\}$.

Termination. Let d be the longest (finite) depth of $\mathsf{T}_{L_{\mathfrak{F}},T_{\mathfrak{F}}(\mathcal{T}^*_\alpha),\Sigma_{\mathcal{T}^*}}$ for $\alpha \in (0,1]_p$. The depth of the computation tree of $(L_{\mathfrak{F}_\pi}, T_{\mathfrak{F}_\pi}(\mathcal{T}))$ with input $\Sigma_{\mathcal{T}}$ is bounded by d times the number of values in $(0,1]_p$ plus a constant factor that comprises the computation needed to rewrite queries asked by $L_{\mathfrak{F}}$ and the final computation of $\mathcal{H} \equiv \mathcal{T}$. Thus, we can transfer learnability of $\mathfrak{F}$ (with only membership queries) to $\mathfrak{F}^p_\pi$.

($\Leftarrow$) We now show the other direction. Let $\mathcal{K} \in \mathcal{L}$ be the target, and assume $T_{\mathfrak{F}}$ is a terminating teacher. We describe the action of a learner $L_{\mathfrak{F}}$ such that $\mathsf{T}_{L_{\mathfrak{F}},T_{\mathfrak{F}}(\mathcal{K}),\Sigma_{\mathcal{K}}}$ has a finite depth and that $\mathcal{H} \in (L_{\mathfrak{F}}, T_{\mathfrak{F}}(\mathcal{K}))(\Sigma_{\mathcal{K}})$ satisfies $\mathcal{H} \equiv \mathcal{K}$. We know $\mathfrak{F}_\pi$ is learnable, therefore there is a terminating teacher $T_{\mathfrak{F}_\pi}$ and a learner $L_{\mathfrak{F}_\mathcal{T}}$ for $\mathfrak{F}_\pi$ such that for every $\mathcal{T} \in \mathcal{L}_\pi$, $\mathsf{T}_{L_{\mathfrak{F}_\pi},T_{\mathfrak{F}_\pi}(\mathcal{T}),\Sigma_{\mathcal{T}}}$ is finite and $\mathcal{H} \in (L_{\mathfrak{F}_\pi}, T_{\mathfrak{F}_\pi}(\mathcal{T}))(\Sigma_{\mathcal{T}})$ implies $\mathcal{H} \equiv \mathcal{T}$. By definition of $\mathfrak{F}_\pi$, we have that $\mathcal{T} = \{(\phi, 1) \mid \phi \in \mathcal{K}\} \in \mathcal{L}_\pi$.

Description of $L_{\mathfrak{F}}$'s steps. $L_{\mathfrak{F}}$ repeats every step that the learner $L_{\mathfrak{F}_\pi}$ performs in paths in $\mathsf{T}_{L_{\mathfrak{F}_\pi},T_{\mathfrak{F}_\pi}(\mathcal{T}),\Sigma_{\mathcal{T}}}$. At the i-th call to $\mathsf{MQ}_{\mathfrak{F}_\pi,\mathcal{T}}$ with input (ϕ,α) in a path p in $\mathsf{T}_{L_{\mathfrak{F}_\pi},T_{\mathfrak{F}_\pi}(\mathcal{T}),\Sigma_{\mathcal{T}}}$, $L_{\mathfrak{F}}$ calls $\mathsf{MQ}_{\mathfrak{F},\mathcal{K}}$ with ϕ as input. By Lemma 4.10, $\mathsf{MQ}_{\mathfrak{F}^p_\pi,\mathcal{T}}(\phi,\alpha) = \mathsf{MQ}_{\mathfrak{F},\mathcal{K}}(\phi)$, therefore $L_{\mathfrak{F}}$ is able to perform every step in p and compute $\mathcal{H} \in (L_{\mathfrak{F}^p_\pi}, T_{\mathfrak{F}^p_\pi}(\mathcal{T}))(\Sigma_{\mathcal{T}})$ that satisfies $\mathcal{H} \equiv \mathcal{T}$. As $\mathcal{H}^* \equiv \mathcal{T}^* = \mathcal{K}$, we have that $\mathcal{H}^*$ is as required.

Termination. Since $\mathsf{T}_{L_{\mathfrak{F}_\pi},T_{\mathfrak{F}_\pi}(\mathcal{T}),\Sigma_{\mathcal{T}}}$ is finite, also $\mathsf{T}_{L_{\mathfrak{F}},T_{\mathfrak{F}}(\mathcal{K}),\Sigma_{\mathcal{K}}}$ is finite. This is because $L_{\mathfrak{F}}$ copies every step of a path in $\mathsf{T}_{L_{\mathfrak{F}_\pi},T_{\mathfrak{F}_\pi}(\mathcal{T}),\Sigma_{\mathcal{T}}}$ and each query (ϕ,α) asked by $L_{\mathfrak{F}_\pi}$ can be translated in polynomial time with respect to its size. Thus, we can transfer learnability of $\mathfrak{F}^p_\pi$ (with only membership queries) to $\mathfrak{F}$. □

If, e.g., $\mathsf{MQ}_{\mathfrak{F}_\pi,t}((\phi,0.01))$ = 'yes', $\mathsf{MQ}_{\mathfrak{F}_\pi,t}((\phi,0.02))$ = 'no', and the precision of the target is 2, then $\mathsf{val}(\phi,t) = 0.01$. So, knowing the precision is important for learning with membership queries only. Otherwise, it will not be possible to determine whether more digits are needed to find the valuation of the formula or whether it has already been found. The direction from possibilistic to the classical setting in Theorem 4.11 normally holds. It is easy to see it because classical formulas are a special case of possibilistic formulas which have been assigned a necessity value of 1. Therefore, an algorithm that learns possibilistic hypotheses can learn also hypotheses with only formulas with necessity degree 1 (that are then dropped to represent any classical hypothesis). The difficulty of showing this result is to simulate possibilistic queries using a classical oracle. More in detail, the answer of a membership query in the possibilistic settings carries more information than the answer of a membership query in the classical settings. Membership queries give more power to the learner to obtain new information. The same argument holds when showing learnability and polynomial time learnability results (Section 4.3) with subset and superset queries.

Corollary 4.12 states that having a terminating teacher that answers equivalence queries is a sufficient condition for learnability. The argument is similar to Theorem 2.13 (Chapter 2) adapted with learning system terminology. The other direction trivially holds by definition.

Corollary 4.12. *Let $\mathfrak{F}$ be an FO learning framework. $\mathfrak{F}$ has a terminating teacher that answers equivalence queries iff $\mathfrak{F}$ is in* $\textsc{El}(\mathsf{EQ})$.

If equivalence queries are allowed then a learner can build a hypothesis equivalent to the target *without knowing the precision* in advance (Theorem 4.13).

Theorem 4.13. *If an FO learning framework $\mathfrak{F}$ has a terminating teacher that answers equivalence queries, then $\mathfrak{F}_\pi$ is in* $\textsc{El}(\mathsf{EQ})$.

Proof. If an FO learning framework $\mathfrak{F}$ has a terminating teacher that answers equivalence queries then, by Remark 4.1, there is a terminating teacher $T_{\mathfrak{F}_\pi}$ that answers equivalence queries posed by a naive learner $L_{\mathfrak{F}_\pi}$ for the learning system $(L_{\mathfrak{F}_\pi}, T_{\mathfrak{F}_\pi})$. By Corollary 4.12, the statement holds. The learner does not know the size of the target in advance but it can estimate it to be n, ask all possible hypotheses of this size, then increase to $n+1$, and so on. In this case, it also needs to estimate the precision of the target and increase it as it navigates the search space. As the precision of the target is finite, eventually $L_{\mathfrak{F}_\pi}$ also halts and outputs an equivalent hypothesis. □

If both membership and equivalence query oracles are available, learnability is guaranteed by the previous theorem (even when the precision of the target is unknown) and the following corollary holds.

Corollary 4.14. *If an FO learning framework $\mathfrak{F}$ has a terminating teacher that answers membership and equivalence queries, then $\mathfrak{F}_\pi$ is in* $\textsc{El}(\mathsf{MQ},\mathsf{EQ})$.

We conclude this section by arguing about learnability when the learner can ask subset or superset queries. If a learner uses one of these two queries, in general, learnability is not guaranteed (Theorem 4.15). If the precision of the target is not known and the learner can ask only one of these queries, it is impossible for the learner to know when to stop increasing the precision of the valuation of the formulas in the built hypothesis, even when there is a positive learnability result in the classical setting.

Theorem 4.15. *For every non-trivial FO learning framework $\mathfrak{F}$, we have that $\mathfrak{F}_\pi$ is neither in* $\textsc{El}(\mathsf{SpQ})$ *nor in* $\textsc{El}(\mathsf{SbQ})$.

If the precision of the target is known, learnability can be transferred, similarly to the case where the learner can ask only membership queries. We show with Theorem 4.16 the learnability result when only subset queries can be asked.

Theorem 4.16. *Let $\mathfrak{F} = (\mathcal{E}, \mathcal{L}, \mu)$ be a safe FO learning framework and for any $p \in \mathbb{N}^+$ let $\mathfrak{F}^p_\pi = (\mathcal{E}_\pi, \mathcal{L}^p_\pi, \mu)$ be its possibilistic extension. $\mathfrak{F}$ is in* EL(SbQ) *iff $\mathfrak{F}^p_\pi$ is in* EL(SbQ).

And we also show in Theorem 4.17 the learnability result when the learner has access only to superset query oracle.

Theorem 4.17. *Let $\mathfrak{F} = (\mathcal{E}, \mathcal{L}, \mu)$ be a safe FO learning framework and, for any $p \in \mathbb{N}^+$, let $\mathfrak{F}^p_\pi = (\mathcal{E}_\pi, \mathcal{L}^p_\pi, \mu)$ be its possibilistic extension. $\mathfrak{F}$ is in* EL(SpQ) *iff $\mathfrak{F}^p_\pi$ is in* EL(SpQ).

It is possible to simulate an equivalence query with input a hypothesis $\mathcal{H}$ by first asking a superset and then a subset query with input $\mathcal{H}$. If both queries returned 'yes', then $\mathcal{H}$ is logically equivalent to the target. Otherwise a counterexample is returned. As a consequence of Theorem 4.13, learnability is guaranteed (Corollary 4.18) if both superset and subset queries can be asked.

Corollary 4.18. *Let $\mathfrak{F}$ be an FO learning framework that has a terminating teacher that answers subset and superset queries, then $\mathfrak{F}_\pi$ is in* EL(SbQ,SpQ).

In Appendix A.2, the reader can find all the omitted proofs. In the next section we focus on polynomial time learning transferability results.

4.3 Polynomial Time Reduction

In this section we investigate whether, in the exact learning model, results showing that an FO learning framework is polynomial time learnable can be transferred to its possibilistic extensions and vice-versa. The general idea behind every proof is to assume that a learning framework $\mathfrak{F}$ is polynomial time learnable and use one (or multiple) learner(s) for $\mathfrak{F}$ to show that another learning framework is polynomial time learnable.

We start by showing a general result when both membership and equivalence queries can be asked. Theorem 4.19 states the transferability of ELP(MQ,EQ) membership from the possibilistic extension $\mathfrak{F}_\pi$ of an FO learning framework $\mathfrak{F}$ to its classical setting. For example if $\mathfrak{F} = (\mathcal{E}, \mathcal{L}, \mu)$ is a propositional learning framework, the idea is that if we can learn the hypothesis $\mathcal{H} = \{(\mathsf{v}_1 \wedge \mathsf{v}_2, 1)\} \in \mathcal{L}_\pi$, then we have learned $\mathcal{H}^* = \{\mathsf{v}_1 \wedge \mathsf{v}_2\}$. Recall that as discussed in the previous section, the challenge in this direction comes

from the fact that the answer of a membership query $\mathsf{MQ}_{\mathfrak{F}_\pi,\mathcal{T}}$ carries more information than an answer given by $\mathsf{MQ}_{\mathfrak{F},\mathcal{T}}$.

Theorem 4.19. *Let $\mathfrak{F}$ be an FO learning framework. If its possibilistic extension $\mathfrak{F}_\pi$ is in* ELP(MQ,EQ)*, then $\mathfrak{F}$ is in* ELP(MQ,EQ)*.*

The converse of Theorem 4.19 does not hold as shown by Theorem 4.21. Simple FO learning frameworks can become difficult to learn when extended with possibilistic valuations because algorithms also have to deal with multiple valuations (Example 4.20).

Example 4.20. Let $\mathfrak{F}^\perp = (\mathcal{E}, \mathcal{L}^\perp)$ be a propositional learning framework with $\mathcal{L}^\perp = \{\{\mathsf{v}, \neg\mathsf{v}\}\}$. $\mathfrak{F}^\perp$ is learnable with only membership queries in constant time because there is a learner L that can directly form an inconsistent hypothesis. The possibilistic extension $\mathfrak{F}^\perp_\pi = (\mathcal{E}_\pi, \mathcal{L}^\perp_\pi)$ of $\mathfrak{F}^\perp$ allows for partial inconsistency as explained in Subsection 2.3 (Chapter 2). Therefore, in $\mathcal{L}^\perp_\pi$ there are hypotheses such as $\{(\mathsf{v}, 0.3282), (\neg\mathsf{v}, 0.701)\} \in \mathcal{L}^\perp_\pi$ that are partially inconsistent. By Theorem 4.8, $\mathfrak{F}^\perp_\pi$ is not learnable. As a consequence, we cannot transfer learnability from the learning framework $\mathfrak{F}^\perp$ to $\mathfrak{F}^\perp_\pi$. ◁

The FO learning framework $\mathfrak{F}^\perp$ in Example 4.20 (or in the proof of Theorem 4.21) is not safe (see definition in Section 2.4 in Chapter 2) because, for $\mathcal{H} \not\subseteq \{\phi, \neg\phi\}$ we have $\mathcal{H} \in \mathcal{L}^\perp$ with $(\mathcal{H} \setminus \{\phi, \neg\phi\}) \notin \mathcal{L}^\perp$. Intuitively, non-safe learning frameworks allow cases in which the target is easy to learn if we aim at learning the *whole* target, but they require a more complex algorithm when only learning a *subset* of it.

Theorem 4.21. *There is an FO learning framework $\mathfrak{F}$ such that $\mathfrak{F}$ is in* ELP(MQ,EQ)*, but $\mathfrak{F}_\pi = (\mathcal{E}_\pi, \mathcal{L}_\pi, \mu_\pi)$ is not in* ELP(MQ,EQ)*.*

Proof. Let $\mathfrak{F} = (\mathcal{E}, \mathcal{L}, \mu)$ be an FO learning framework that is *not* in ELP(MQ,EQ). Such $\mathfrak{F}$ exists, one can consider, for instance, the $\mathcal{EL}$ learning framework [Konev et al., 2018, Theorem 68][1]. We use $\mathfrak{F}$ to define the learning framework $\mathfrak{F}^\perp = (\mathcal{E}, \mathcal{L}^\perp)$ where $\mathcal{L}^\perp = \{\mathcal{H} \cup \{\phi, \neg\phi\} \mid \mathcal{H} \in \mathcal{L}\}$ for a fixed but arbitrary non-trivial FO formula ϕ. Even though $\mathfrak{F}$ is not learnable in polynomial time, $\mathfrak{F}^\perp$ is. The learner $L_{\mathfrak{F}^\perp}$ of the learning system $(L_{\mathfrak{F}^\perp}, T_{\mathfrak{F}^\perp})$ can learn any $\mathcal{H} \in \mathcal{L}^\perp$ by returning the hypothesis $\{\perp\}$ (in constant time). Assume that $\mathfrak{F}^\perp_\pi = (\mathcal{E}_\pi, \mathcal{L}^\perp_\pi)$ is in ELP(MQ,EQ). This means that for every target $\mathcal{T} \in \mathcal{L}^\perp_\pi$ we can obtain in polynomial time a hypothesis $\mathcal{H} \in (L_{\mathfrak{F}^\perp_\pi}, T_{\mathfrak{F}^\perp_\pi})(\Sigma_\mathcal{T})$ such that $\mathcal{H} \equiv \mathcal{T}$. By construction, for every $\mathcal{T} \in \mathcal{L}$ there is $\mathcal{K} \in \mathcal{L}^\perp_\pi$ such that $\mathcal{T} \equiv \mathcal{K}^*_{\mathsf{inc}(\mathcal{K})}$. By learning $\mathcal{H}$ such that $\mathcal{H} \equiv \mathcal{K}$ we have also learned a hypothesis $\mathcal{H}$ such that $\mathcal{H}^*_{\mathsf{inc}(\mathcal{H})} \equiv \mathcal{T}$. By Theorem 4.19, $\mathfrak{F} \in$ ELP(MQ,EQ), which contradicts our assumption that this is

[1] Non-polynomial query learnability is proven in [Konev et al., 2018, Theorem 68], which implies non-polynomial time learnability.

not the case. Therefore, we have found an FO learning framework $\mathfrak{F}^{\perp}$ that is is in ELP(MQ,EQ) but its respective possibilistic extension $\mathfrak{F}_{\pi}^{\perp}$ is not in ELP(MQ,EQ). □

The remaining section is divided in five parts, in the following subsection we prove transferability when only membership queries can be asked. In the next subsections we consider the case when only equivalence queries, only superset, only subset queries, both membership and equivalence queries, or both superset and subset queries can be asked. For the rest of this section, we focus on FO learning frameworks that are safe[2]. The assumption is only used while proving transferability from classical to possibilistic learning frameworks.

Reduction With Membership Queries

The first transferability result we present is for the case in which the learner has access to only membership queries. Before showing the reduction, we present Algorithm 6, which computes the highest valuation, up to a predefined precision p, of a formula ϕ entailed by the target $\mathcal{T}$. That is, β such that $\downarrow_p(\beta) = \mathsf{val}(\phi, \mathcal{T})$. The checks $\mathcal{T} \models (\phi, \beta)$ in Algorithm 6 are implemented with calls to the oracle $\mathsf{MQ}_{\mathfrak{F}_\pi, \mathcal{T}}$, i.e., if the learner wants to check $\mathcal{T} \models (\phi, \beta)$, it calls $\mathsf{MQ}_{\mathfrak{F}_\pi, \mathcal{T}}$ with (ϕ, β) as input.

Algorithm 6: Finding the valuation of a formula.

```
1:  Input: precision p ∈ ℕ⁺, a formula φ
2:  Output:β such that ↓_p(β) = val(φ, T).
3:  min ← 0, max ← 1
4:  while (max − min) > 10^−p do
5:     Compute β = ↓_p((min+max)/2)                    // β is always > 0
6:     if T ⊨ (φ, β)                                   // membership queries
       then
7:        min ← β
8:     else
9:        max ← β
10:    end if
11: end while
12: if T ⊨ (φ, max)                                    // membership queries
    then
13:    return max
14: end if
15: return min                                         // value 0 can be returned
```

The value β computed in Line 5 is always greater than 0 because $max - min > 10^{-p}$ in

[2]Most learning settings considered in the literature are safe.

the main while loop. This is important because the value 0 is not a valid valuation for a (necessity valued) possibilistic formula. Also, the check in Line 11 calls $\mathsf{MQ}_{\mathfrak{F}_\pi,\mathcal{T}}$ with a formula with valuation at least 10^{-p} as input. Lemma 4.22 states the correctness of Algorithm 6 and the fact that it runs in polynomial time w.r.t. the size of the input.

Lemma 4.22. *Let $\mathfrak{F}_\pi = (\mathcal{E}_\pi, \mathcal{L}_\pi, \mu_\pi)$ be a possibilistic learning framework and let $\mathcal{T} \in \mathcal{L}_\pi$ be the target. Algorithm 6, with input a precision $p \in \mathbb{N}^+$ and $\phi \in \mathcal{E}_\pi$, runs in polynomial time in p and $|\phi|$ and outputs β such that $\downarrow_p(\beta) = \mathsf{val}(\phi, \mathcal{T})$.*

Proof. By Point 4 of Proposition 4.7, Algorithm 6 can determine β such that $\downarrow_p(\beta) = \mathsf{val}(\phi, \mathcal{T})$ by performing a binary search on the interval of numbers $[0,1]_p$. The computation of β in Line 5 requires polynomial time w.r.t. p. The check $\mathcal{T} \models (\phi, \beta)$ in Lines 6,12 consists of asking a membership query with input (ϕ, β). It is positive if the answer is 'yes', otherwise it is negative. We recall that to ask a membership query, the learner writes ϕ and β on the communication tape. This step can be done in polynomial time w.r.t. $|\phi|$ and p. A membership query counts as a step of computation and each loop can be computed in polynomial time w.r.t. $|\phi|$ and p. The algorithm performs a binary search on the interval of numbers $[0,1]_p$, therefore the number of iterations is bounded by $log_2[0,1]_p \leq log_2 10^p < 4p$ which is polynomial w.r.t. p. □

A simple run of Algorithm 6 is presented in Example 4.23.

Example 4.23. Let $\mathfrak{F}$ be a propositional learning framework and let $\mathfrak{F}^1_\pi = (\mathcal{E}^1_\pi, \mathcal{L}^1_\pi)$ be its possibilistic extension. Assume the target is $\mathcal{T} = \{(\mathsf{v}, 0.3)\} \in \mathcal{L}^1_\pi$. The learner $L_{\mathfrak{F}^1_\pi}$ of the learning system $(L_{\mathfrak{F}^1_\pi}, T_{\mathfrak{F}^1_\pi})$ can find $\mathsf{val}(\mathsf{v}, \mathcal{T})$ by running Algorithm 6 with input 1 and v. The first membership query asked by $L_{\mathfrak{F}^1_\pi}$ would be with input the possibilistic formula $(\mathsf{v}, 0.5)$, the response would be 'no' and max is set to 0.5. The second query will be $(\mathsf{v}, 0.2)$ with answer 'yes'. After that, $min = 0.2$ and the query will have as input $(\mathsf{v}, 0.3)$ and the answer will be 'yes'. At this point $min = 0.3$ and $max = 0.5$. After the query with input $(\mathsf{v}, 0.4)$, min is set to 0.3 and max to 0.4. Since $max - min \leq 10^{-p}$, the execution will exit from the main loop and the algorithm will return 0.3 after having asked the last query with input $(\mathsf{v}, 0.4)$. ◁

In our next theorem, we show that, for safe FO learning frameworks, polynomial time results with only membership queries can be transferred to their possibilistic extensions if the precision of the target is known (recall that, by Theorem 4.8, we cannot remove this assumption because the learning framework is not learnable).

Theorem 4.24. *Let $\mathfrak{F}$ be a safe FO learning framework. For all $p \in \mathbb{N}^+$, $\mathfrak{F}$ is in* ELP(MQ) *iff $\mathfrak{F}^p_\pi$ is in* ELP(MQ).

Proof. ($\Rightarrow$) To show the transferability of ELP(MQ) membership from $\mathfrak{F}$ to $\mathfrak{F}_\pi$, we use the following claim.

Claim 4.25. *Assume $\mathfrak{F} = (\mathcal{E}, \mathcal{L}, \mu)$ is safe and in* ELP(MQ). *For every $p \in \mathbb{N}^+$ and framework $\mathfrak{F}_\pi^p = (\mathcal{E}_\pi, \mathcal{L}_\pi^p)$ with $\mathcal{T} \in \mathcal{L}_\pi^p$, given a valuation α with $\mathsf{prec}(\alpha) = p$, one can learn $\mathcal{K}_{\overline{\alpha}}^*$ such that $\mathcal{K}_{\overline{\alpha}}^* \equiv \mathcal{T}_{\overline{\alpha}}^*$ in time polynomial w.r.t. $|\mathcal{T}|$ with only membership queries.*

Proof. $\mathfrak{F}$ in ELP(MQ) implies that there is a learning system $(L_\mathfrak{F}, T_\mathfrak{F})$ such that for any $\mathcal{K} \in \mathcal{L}$, $\mathsf{T}_{L_\mathfrak{F}, T_\mathfrak{F}(\mathcal{K}), \Sigma_\mathcal{K}}$ has a finite depth, and $\mathcal{H} \in (L_\mathfrak{F}, T_\mathfrak{F}(\mathcal{K}))(\Sigma_\mathcal{K})$ implies $\mathcal{H} \equiv \mathcal{K}$. Since $\mathfrak{F}$ is safe, we have that $\mathcal{T}_{\overline{\alpha}}^* \in \mathcal{L}$. We consider the computation of the learning system $(L_{\mathfrak{F}_\pi}, T_{\mathfrak{F}_\pi}(\mathcal{T}))$ on input $\Sigma_\mathcal{T}$ where $L_{\mathfrak{F}_\pi}$ performs every step done by $L_\mathfrak{F}$ in an arbitrary path p in $\mathsf{T}_{L_\mathfrak{F}, T_\mathfrak{F}(\mathcal{T}_{\overline{\alpha}}^*), \Sigma_{\mathcal{T}^*}}$. $\Sigma_{\mathcal{T}^*}$ is computed from $\Sigma_\mathcal{T}$ (Remark 4.2). At the i-th call to $\mathsf{MQ}_{\mathfrak{F}, \mathcal{T}_{\overline{\alpha}}^*}$ with ϕ as input in p, $L_{\mathfrak{F}_\pi}$ calls $\mathsf{MQ}_{\mathfrak{F}_\pi, \mathcal{T}}$ with $(\phi, \alpha + 10^{-p})$ as input. By Point 3 of Proposition 4.7, $\mathsf{MQ}_{\mathfrak{F}, \mathcal{T}_{\overline{\alpha}}^*}(\phi) = \mathsf{MQ}_{\mathfrak{F}_\pi, \mathcal{T}}(\phi, \alpha + 10^{-p})$. This query simulation requires polynomially many steps with respect to the size of the query. In this way, $L_{\mathfrak{F}_\pi}$ will compute in polynomially many steps w.r.t. the number of steps made by $L_{\mathfrak{F}_\pi}$ in p a hypothesis $\mathcal{K}_{\overline{\alpha}}^*$ such that $\mathcal{K}_{\overline{\alpha}}^* \equiv \mathcal{T}_{\overline{\alpha}}^*$. The number of steps performed by $L_\mathfrak{F}$ in p is polynomial with respect to $|\mathcal{T}_{\overline{\alpha}}^*|$, therefore the claim holds. □

Given the learning system $(L_{\mathfrak{F}_\pi}, T_{\mathfrak{F}_\pi}(\mathcal{T}))$ on input $\Sigma_\mathcal{T}$, the learner $L_{\mathfrak{F}_\pi}$ can initialise two helper variables $\gamma := 0$ and $S := \emptyset$. By Claim 4.25, it can find in polynomial time w.r.t. $|\mathcal{T}|$ a hypothesis $\mathcal{K}_{\overline{\gamma}}^*$ such that $\mathcal{K}_{\overline{\gamma}}^* \equiv \mathcal{T}_{\overline{\gamma}}^*$. For every $\phi \in \mathcal{K}_{\overline{\gamma}}^*$, we run Algorithm 6 with $p = \mathsf{prec}(\mathcal{T})$ and ϕ as input to find $\mathsf{val}(\phi, \mathcal{T})$. In this way, by Point 5 of Proposition 4.7 and Lemma 4.22, $L_{\mathfrak{F}_\pi}$ identifies in polynomial time w.r.t. $|\mathcal{T}|$ the smallest $\beta \in \mathcal{T}^v \cup \{1\}$ such that $\mathcal{K}_{\overline{\gamma}}^* \equiv \mathcal{T}_\beta^*$. $L_{\mathfrak{F}_\pi}$ sets $\mathcal{K}_\beta^* := \mathcal{K}_{\overline{\gamma}}^*$ and it adds $\mathcal{K}_\beta^*$ to S. Then, γ is updated to the value β and Claim 4.25 is applied again. For every $\phi \in \mathcal{K}_{\overline{\gamma}}^*$, $L_{\mathfrak{F}_\pi}$ runs Algorithm 6 again with $p = \mathsf{prec}(\mathcal{T})$ and ϕ as input to find $\mathsf{val}(\phi, \mathcal{T})$. This process is repeated until $L_{\mathfrak{F}_\pi}$ finds $\mathcal{K}_{\overline{\gamma}}^* \equiv \emptyset$ or $\gamma + 10^{-p} > 1$. Each time $L_{\mathfrak{F}_\pi}$ runs Algorithm 6, it identifies a higher valuation in $\mathcal{T}^v$. Therefore, this happens at most $|\mathcal{T}^v|$ times. For all $\alpha \in \mathcal{T}^v$, there is $\mathcal{K}_\alpha^* \in S$ that satisfies $\mathcal{K}_\alpha^* \equiv \mathcal{T}_\alpha^*$, therefore, by Lemma 4.9,

$$\mathcal{H} = \bigcup_{\mathcal{K}_\alpha^* \in S} \{(\phi, \alpha) \mid \phi \in \mathcal{K}_\alpha^*\}$$

is such that $\mathcal{H} \equiv \mathcal{T}$. The number of steps that $L_{\mathfrak{F}_\pi}$ makes is bounded by a polynomial w.r.t. $|\mathcal{T}|$, therefore $\mathfrak{F}_\pi$ is in ELP(MQ).

($\Leftarrow$) We now show the transferability of ELP(MQ) membership from $\mathfrak{F}_\pi$ to $\mathfrak{F}$. Let $\mathcal{K} \in \mathcal{L}$ be the target. Since $\mathfrak{F}_\pi$ is in ELP(MQ), for each $\mathcal{T} \in \mathcal{L}_\pi$, there is a learning system $(L_{\mathfrak{F}_\pi}, T_{\mathfrak{F}_\pi}(\mathcal{T}))$ such that $\mathsf{T}_{L_{\mathfrak{F}_\pi}, T_{\mathfrak{F}_\pi}(\mathcal{T}), \Sigma_\mathcal{T}}$ is finite, and $\mathcal{H} \in (L_{\mathfrak{F}_\pi}, T_{\mathfrak{F}_\pi}(\mathcal{T}))(\Sigma_\mathcal{T})$ implies $\mathcal{H} \equiv \mathcal{T}$. By definition of $\mathfrak{F}_\pi$, $\mathcal{T} = \{(\phi, 1) \mid \phi \in \mathcal{K}\} \in \mathcal{L}_\pi$. We consider the computation of

the learning system $(L_{\mathfrak{F}}, T_{\mathfrak{F}}(\mathcal{K}))$ on input $\Sigma_{\mathcal{K}}$ where $L_{\mathfrak{F}}$ performs every step made by $L_{\mathfrak{F}_\pi}$ in an arbitrary path p in $\mathsf{T}_{L_{\mathfrak{F}_\pi}, T_{\mathfrak{F}_\pi}(\mathcal{T}), \Sigma_{\mathcal{T}}}$. $\Sigma_{\mathcal{T}}$ is built from $\Sigma_{\mathcal{K}}$. At the i-th call to $\mathsf{MQ}_{\mathfrak{F}_\pi, \mathcal{T}}$ with input (ϕ, α) in p, $L_{\mathfrak{F}}$ calls $\mathsf{MQ}_{\mathfrak{F}, \mathcal{K}}$ with ϕ as input. The number of steps needed to compute the translation of the query is polynomial with respect to its size. By Lemma 4.10, $\mathsf{MQ}_{\mathfrak{F}_\pi, \mathcal{T}}(\phi, \alpha) = \mathsf{MQ}_{\mathfrak{F}, \mathcal{K}}(\phi)$ and $L_{\mathfrak{F}}$ is able to compute the hypothesis $\mathcal{H}$ such that $\mathcal{H} \equiv \mathcal{T}$. As $\mathcal{H}^* \equiv \mathcal{T}^* = \mathcal{K}$, $\mathcal{H}^*$ is as required. The number of steps that $L_{\mathfrak{F}}$ executes is polynomial w.r.t. the number of steps made by $L_{\mathfrak{F}_\pi}$ in p. As the latter number is polynomial w.r.t. $|\mathcal{T}|$, which is polynomial w.r.t. $|\mathcal{K}|$, $\mathfrak{F}_\pi$ is in $\textsc{ELP}(\mathsf{MQ})$. □

When we want to transfer learnability results from $\mathfrak{F}$ to $\mathfrak{F}_\pi$ it is important to learn one $\mathcal{H}_\alpha$ such that $\mathcal{H}_\alpha \equiv \mathcal{T}_\alpha$ for each $\alpha \in \mathcal{T}^v$, where $\mathcal{T}$ is the target. Example 4.26 provides a case where if the layers of the target are not taken into account, the final generated hypothesis fails to be logically equivalent to the target.

Example 4.26. Let $\mathcal{T} = \{(\mathsf{v}_1 \rightarrow \mathsf{v}_2, 0.3), (\mathsf{v}_1 \rightarrow \mathsf{v}_3, 0.7)\}$. We can use the polynomial time algorithm for propositional Horn [Angluin et al., 1992] (extended to many variables in the consequent). With it we would learn a hypothesis $\mathcal{K}^* = \{\mathsf{v}_1 \rightarrow (\mathsf{v}_2 \wedge \mathsf{v}_3)\} \equiv \mathcal{T}^*$. However, if we build the hypothesis $\mathcal{H} = \{(\phi, \mathsf{val}(\phi, \mathcal{T})) \mid \phi \in \mathcal{K}^*\}$ by asking membership queries, then we would have $\mathcal{H} = \{(\mathsf{v}_1 \rightarrow (\mathsf{v}_2 \wedge \mathsf{v}_3), 0.3)\} \not\equiv \mathcal{T}$. ◁

Learning classical hypothesis separately, allows the learner to find every valuation occurring in the target.

Reduction With Equivalence Queries

The main difficulty in the reduction with only equivalence queries is that we do not know the precision of the target. But under certain conditions, a learner who has access to only equivalence queries sometimes has a way of deducing if a given precision is smaller than the precision of the target (Claim 4.30). To illustrate this idea consider Example 4.27.

Example 4.27. Let $\mathcal{T}$ be the target and assume $\mathcal{T} \models (\mathsf{v}, 0.32)$. Suppose the learner receives the positive counterexample $(\mathsf{v}, 0.32)$. If the precision of $\mathcal{T}$ is 1 then it must be the case that $\mathcal{T} \models (\mathsf{v}, 0.4)$. If the learner also receives the negative counterexample $(\mathsf{v}, 0.33)$, then the precision of $\mathcal{T}$ cannot be 1 because, otherwise, $\mathcal{T} \not\models (\mathsf{v}, 0.3)$, which cannot happen if $\mathcal{T} \models (\mathsf{v}, 0.4)$. This reasoning is justified by Point 4 of Proposition 4.7. Therefore, the precision of $\mathcal{T}$ is larger than 1. ◁

The polynomial time reduction from classical to possibilistic settings presented in this section is difficult to show in the general case without any assumption made on the

learner or the learning framework into consideration (Example 4.28). As illustrated in Example 4.27, one can determine if the estimated precision of the target is too low with equivalence queries. In the proof, the learner can get this information when the search space (of hypotheses with a fixed length) is completely pruned. While showing polynomial time learnability results, we cannot apply the same strategy as in Theorem 4.13. If a learner can receive both positive and negative counterexamples, we cannot know if the counterexample returned by an equivalence query is due to the wrong hypothesis built by the classical learner(s) or the estimated precision. Therefore, while showing transferability results we have to rely on other assumptions. In this section, we assume that the learner is positive bounded (Section 4.1), that is the hypothesis generated by the learner is always logically entailed by the target. So when it receives a negative counterexample, it can use this information to tune the estimated precision of the target.

Example 4.28. Let $L_{\mathfrak{F}_\pi}$ be a learner for $\mathfrak{F}_\pi$ that uses the polynomial time learnability result of a classical learner $L_{\mathfrak{F}}$ for $\mathfrak{F} = (\mathcal{E}, \mathcal{L}, \mu)$ where $\mathcal{L}$ is a conjunction of literals and $\mathcal{E}$ is the set of all conjunctions of literals. We may have the additional information that the target has two valuations: $\alpha_1, \alpha_2 \in (0, 1]$, $\alpha_1 < \alpha_2$; and as explained before (Example 4.26), $L_{\mathfrak{F}_\pi}$ should copy the steps made by $L_{\mathfrak{F}}$ to learn classical hypotheses for each valuation. Let $\mathcal{K}^{\alpha_1} = \{\mathsf{v}_1\}$, $\mathcal{K}^{\alpha_2} = \{\mathsf{v}_2\}$ be two hypotheses built by a possibilistic learner $L_{\mathfrak{F}_\pi}$ according to the steps made by $L_{\mathfrak{F}}$ for each α_i ($L_{\mathfrak{F}}$ may want to initialise hypotheses randomly). $L_{\mathfrak{F}_\pi}$ may ask an equivalence query with input the hypothesis $\mathcal{H} = \bigcup_{i=1}^{2}\{(\phi, \alpha_i) \mid \phi \in \mathcal{K}^{\alpha_i}\}$. If a negative counterexample of the form $(\mathsf{v}_1 \wedge \mathsf{v}_2, \gamma)$ with $\gamma \leq \alpha_1$ is returned, $L_{\mathfrak{F}_\pi}$ does not know how to use the information provided by the counterexample to continue with the steps made by the classical learner because no built classical hypothesis entails $(\mathsf{v}_1 \wedge \mathsf{v}_2)$. In this case, we may receive a counterexample (ϕ, γ) from a call to $\mathsf{EQ}_{\mathfrak{F}_\pi, \mathcal{T}}$ such that ϕ is not the counterexample of any classical hypothesis built so far by any instance of $L_{\mathfrak{F}}$.

The learner can build $\mathcal{K}^{\alpha_1}$ such that $\mathcal{K}^{\alpha_1} \models \mathcal{K}^{\alpha_2}$ by considering formulas in $\mathcal{K}^{\alpha_2}$ not entailed by $\mathcal{K}^{\alpha_1}$ as positive counterexamples when repeating the steps made by $L_{\mathfrak{F}}$. Let $\phi \in \mathcal{K}^{\alpha_2}$ be a formula that is considered as a positive counterexample while building $\mathcal{K}^{\alpha_1}$. Since we cannot assume that $\mathcal{K}^{\alpha_2}$ is entailed by the target, the learner can discover only later that ϕ should have never be returned as a positive counterexample. This is problematic because $L_{\mathfrak{F}}$ (as it is defined) expects the oracle to answer truthfully. ◁

As illustrated in Example 4.28, a negative example (ϕ, α) may refer to entailments coming from the combination of classical hypotheses that alone do not entail ϕ. In the most general case, it is not clear how counterexamples can be used to continue the steps made by the classical learner. For this reason, we limit the transferability result from a classical learning framework $\mathfrak{F}$ to $\mathfrak{F}_\pi$ to the case where there is a positive bounded learner for $\mathfrak{F}$.

Lemma 4.29. *Let $\mathfrak{F}_\pi$ be the possibilistic extension of a safe FO learning framework $\mathfrak{F}$ in* ELP(EQ) *and let $\mathcal{T} \in \mathcal{L}_\pi$ be the target. Assume there is a positive bounded learner for $\mathfrak{F}$. Then, given $p \in \mathbb{N}^+$, one can determine that $p < \mathsf{prec}(\mathcal{T})$ or compute $\mathcal{H} \in \mathcal{L}_\pi$ such that $\mathcal{H} \equiv \mathcal{T}$, in polynomial time with respect to $|\mathcal{T}|$, p, and the largest counterexample seen so far, by asking only equivalence queries.*

Proof. $\mathfrak{F} = (\mathcal{E}, \mathcal{L}, \mu)$ in ELP(EQ) implies that there is a learning system $(L_\mathfrak{F}, T_\mathfrak{F})$ such that for any $\mathcal{K} \in \mathcal{L}$, $L_\mathfrak{F}$ calls only $\mathsf{EQ}_{\mathfrak{F},\mathcal{K}}$, and $\mathcal{H} \in (L_\mathfrak{F}, T_\mathfrak{F})(\Sigma_\mathcal{K})$ implies $\mathcal{H} \equiv \mathcal{K}$. Since $\mathfrak{F}$ is safe, we have that $\mathcal{T}^*_\alpha \in \mathcal{L}$ for any $\alpha \in (0,1]_p$. Let $\mathfrak{F}_\pi = (\mathcal{E}_\pi, \mathcal{L}_\pi, \mu_\pi)$ be the possibilistic extension of $\mathfrak{F}$. We consider the learning system $(L_{\mathfrak{F}_\pi}, T_{\mathfrak{F}_\pi}(\mathcal{T}))$ and we describe the steps of the learner $L_{\mathfrak{F}_\pi}$ that attempts to learn $\mathcal{T}$ on input $\Sigma_\mathcal{T}$. To support our argument, we show the following claims. For a number $p \in \mathbb{R}$, recall the definition of $\uparrow_p(\phi)$ and $\downarrow_p(\phi)$ at the beginning of Section 4.2.

Claim 4.30. *For an unknown $\mathcal{T} \in \mathcal{L}_\pi$, let two finite sets P and N be such that $\mathsf{P} \subseteq \mu_\pi(\mathcal{T})$ and $\mathsf{N} \subseteq \mathcal{E}_\pi \setminus \mu_\pi(\mathcal{T})$, and let $p \in \mathbb{N}^+$. If there is $(\psi, \gamma) \in \mathsf{N}$ such that $\{(\phi, \uparrow_p(\gamma)) \mid (\phi, \gamma) \in \mathsf{P}\} \models (\psi, \gamma)$, then $p < \mathsf{prec}(\mathcal{T})$. This check can be done in polynomial time w.r.t. $|\mathsf{P}|+|\mathsf{N}|$.*

Proof. We know that for every $e \in \mathsf{P}$, $\mathcal{T} \models e$ and that for every $e \in \mathsf{N}$, $\mathcal{T} \not\models e$. If we assume $p \geq \mathsf{prec}(\mathcal{T})$, we need to conclude that there is $(\phi, \gamma) \in \mathsf{N}$ such that $\mathcal{T} \models (\phi, \gamma)$, contradiction. Therefore, when this condition applies, we know that $p < \mathsf{prec}(\mathcal{T})$. We can make this check by looping for every different valuation in $\{\uparrow_p(\gamma) \mid (\phi, \gamma) \in \mathsf{P}\}$ and every element in N. The result follows because of Remark 4.4. □

Claim 4.31. *Let $\mathcal{T} \in \mathcal{L}_\pi$. If for $0 < \gamma < \delta \leq 1$, $\mathcal{T} \models (\phi, \gamma)$, and $\mathcal{T} \not\models (\phi, \delta)$ then there is $\alpha \in \mathcal{T}^v$ with $\gamma \leq \alpha < \delta$.*

Proof. By Points 4 and 5 of Proposition 4.7, $\gamma \leq \mathsf{val}(\phi, t)$, $\mathcal{T} \models (\phi, \mathsf{val}(\phi, \mathcal{T}))$ and $\mathsf{val}(\phi, \mathcal{T}) \in \mathcal{T}^v$. Since $\mathcal{T} \not\models (\phi, \delta)$, by Point 4 of Proposition 4.7, we have $\mathsf{val}(\phi, \mathcal{T}) < \delta$. Therefore, there is $\alpha = \mathsf{val}(\phi, \mathcal{T})$ such that $\alpha \in \mathcal{T}^v$ and $\gamma \leq \alpha < \delta$. □

Let P^i and N^i be respectively the sets of positive and negative counterexamples received so far after the i-th call to $\mathsf{EQ}_{\mathfrak{F}_\pi,\mathcal{T}}$. $L_{\mathfrak{F}_\pi}$ calls $\mathsf{EQ}_{\mathfrak{F}_\pi,\mathcal{T}}$ with input $\emptyset$ ($\emptyset \in \mathcal{L}_\pi$ because $\mathfrak{F}$ is safe). If the target is not the empty set, $L_{\mathfrak{F}_\pi}$ receives a positive counterexample e ($\mathsf{P}^1 = \mathsf{P}^0 \cup \{e\}, \mathsf{N}^1 = \mathsf{N}^0$), otherwise $L_{\mathfrak{F}_\pi}$ receives 'yes' and it terminates. We denote by $\mathcal{V}^i_p$ the set of valuations $\{\uparrow_p(\gamma) \mid (\phi, \gamma) \in \mathsf{P}^i\}$.

Asking an equivalence query. Assume $L_{\mathfrak{F}_\pi}$ has asked the i-th query, $i \geq 0$. For each $\alpha \in \mathcal{V}^i_p$, the learner $L_{\mathfrak{F}_\pi}$ performs every step that $L_\mathfrak{F}$ makes in a path p_α in $\mathsf{T}_{L_\mathfrak{F}, T_\mathfrak{F}(\mathcal{T}^*_\alpha), \Sigma_{\mathcal{T}^*}}$ (Remark 4.2). Whenever there is a call to $\mathsf{EQ}_{\mathfrak{F},\mathcal{T}^*_\alpha}$ with input $\mathcal{K}^\alpha$ in p_α, $L_{\mathfrak{F}_\pi}$ finds a positive

counterexample $(\phi, \gamma) \in \mathsf{P}^i$ (if any) for $\mathcal{K}^\alpha$ with $\alpha \leq \gamma$ and continues the computation in path p. By Points 3 and 4 of Proposition 4.7, each counterexample found in this way satisfies $\mathcal{K}^\alpha \not\models \phi$ and $\mathcal{T}^*_\alpha \models \phi$. For $\alpha \in \mathcal{V}^i_p$, whenever $L_{\mathfrak{F}_\pi}$ reaches the configuration in the associated computation path where $L_{\mathfrak{F}}$ stops in query state with input $\mathcal{K}^{\alpha,i}$ and no counterexample can be found in P^i, $L_{\mathfrak{F}_\pi}$ creates the hypothesis

$$\mathcal{H}^i = \bigcup_{\alpha \in \mathcal{V}^i_p} \{(\phi, mid(\alpha)^i) \mid \phi \in \mathcal{K}^{\alpha,i}\}$$

where

$$mid(\alpha)^i = \uparrow_p \left(\frac{\alpha + end(\alpha)^i}{2} \right)$$

$$end(\alpha)^i = min(\{\downarrow_p(\delta) \mid (\phi, \delta) \in \mathsf{N}^i \text{ and } \mathcal{K}^{\alpha,i} \models \phi\} \cup \{1\}),$$

and $L_{\mathfrak{F}_\pi}$ calls $\mathsf{EQ}_{\mathfrak{F}_\pi,t}$ with input $\mathcal{H}^i$. If the answer is 'yes', $L_{\mathfrak{F}_\pi}$ stops and outputs $\mathcal{H}^i$. Upon receiving a counterexample, $L_{\mathfrak{F}_\pi}$ creates (or updates) P^{i+1}, N^{i+1} and checks by Claim 4.30 if $p < \mathsf{prec}(\mathcal{T})$. If it is the case, it stops. Note that the possibilistic learner is not positive bounded because valuations attached to formulas are guessed with a higher value. Negative counterexamples are returned to signal that the estimated valuation is too high for a formula.

If the counterexample received (ϕ, γ) is positive, $L_{\mathfrak{F}_\pi}$ resumes the computation of all paths p_β such that $\mathcal{K}^{\beta,i} \not\models \phi$ with $\uparrow_p(\gamma) \geq \beta$ (recall Remark 4.4) and starts simulating all the steps that $L_{\mathfrak{F}}$ makes in a path $\mathsf{p}_{\uparrow_p(\gamma)}$ in $\mathsf{T}_{L_{\mathfrak{F}}, T_{\mathfrak{F}}(\mathcal{T}^*_{\uparrow_p(\gamma)}), \Sigma_{\mathcal{T}^*}}$ as described (because $\uparrow_p(\gamma) \in \mathsf{V}^{i+1}$). When for all $\alpha \in \mathcal{V}^{i+1}_p$, the learner $L_{\mathfrak{F}_\pi}$ reaches the configuration in p_α where $L_{\mathfrak{F}}$ stops in query state with input $\mathcal{K}^{\alpha,i+1}$ and no counterexample can be found in P^{i+1}, the learner $L_{\mathfrak{F}_\pi}$ creates $\mathcal{H}^{i+1}$ and calls again $\mathsf{EQ}_{\mathfrak{F}_\pi,\mathcal{T}}$ with input $\mathcal{H}^{i+1}$ until it receives the answer 'yes'. The hypotheses $\mathcal{K}^{\beta,i+1}$ written in the communication tape in paths p_α that are not resumed are not updated, that is they remain the same in the next iteration $\mathcal{K}^{\beta,i+1} := \mathcal{K}^{\beta,i}$.

Each iteration i is computed in polynomial time with respect to the size of $|\mathsf{P}|^i$, $|\mathsf{N}|^i$, and p (see also Claim 4.30). The maximum number of iterations corresponds to the maximum number of elements in the set P of positive counterexamples and N of negative counterexamples. In the following, we argue about their size and we show that they are polynomial with respect to $|\mathcal{T}|$, p, and the largest counterexample seen so far.

Size of P. Let i be the number of equivalence queries asked so far. A positive counterexample (ϕ, γ) is received when $\mathsf{val}(\phi, \mathcal{H}^i) < \gamma \leq \mathsf{val}(\phi, \mathcal{T})$. If $\mathsf{val}(\phi, \mathcal{H}^i) = 0$, by monotonicity and by Point 3 of Proposition 4.7, no $\mathcal{K}^{\alpha,i}$ entails ϕ for every $\alpha \in \mathcal{V}^i_p$. At

iteration $i+1$, there will be the smallest $\beta \in \mathcal{V}_p^{i+1}$, such that $\mathcal{K}^{\beta,i} \models \phi$. Since $L_{\mathfrak{F}}$ is a polynomial time learner for $\mathfrak{F}$, this case can happen at most $g(|\mathcal{T}_\beta^*|, |e|)$ times where g is a polynomial and e is the largest counterexample seen so far. Indeed, after $g(|\mathcal{T}_\beta^*|, |e|)$ counterexamples falling in this case, for the smallest $\beta \in \mathcal{V}_p^{i+1}$, we have $\mathcal{K}^{\beta,i} \models \mathcal{T}^*$, hence $\mathcal{K}^{\beta,i} \equiv \mathcal{T}^*$ because $L_{\mathfrak{F}}$ is positive bounded. If $0 < \mathsf{val}(\phi, \mathcal{H}^i) < \gamma \leq \mathsf{val}(\phi, \mathcal{T})$, by construction and by Point 5 of Proposition 4.7, we have that there is $\beta \in \mathcal{V}_p^i = \{\uparrow_p(\gamma) \mid \gamma \in \mathsf{P}^i\}$ such that $mid(\beta)^i = \mathsf{val}(\phi, \mathcal{H}^i)$ and $\gamma > \uparrow_p\left(\frac{\beta + end(\beta)^i}{2}\right)$. At the next iteration $\mathsf{val}(\phi, \mathcal{H}^{i+1})$ will be equal to $\uparrow_p\left(\frac{\gamma + end(\beta)^i}{2}\right)$ or greater than $end(\beta)^i$. This means that this case can happen at most $log_2(\beta, end(\beta)^i) \leq log_2(0,1]_p \leq log_2 10^p < 4p$ times. By Claim 4.31 and since for every $\alpha, \beta \in \mathcal{V}_p^i$ with $\alpha < \beta$, we have that $\mathcal{K}^{\alpha,i} \models \mathcal{K}^{\beta,i}$,[3] there can be at most $|\mathcal{T}^v|$ different $end(\beta)^i$. Therefore, a positive counterexample is returned a number of times at most $g(|\mathcal{T}|, |e|) \cdot |\mathcal{T}^v| \cdot 4p$.

Size of N. A negative counterexample (ϕ, γ) is returned when $\mathsf{val}(\phi, \mathcal{H}^i) > \mathsf{val}(\phi, \mathcal{T})$. For every $\alpha \in \mathcal{V}_p^i$, $\alpha \geq \gamma$, by construction, $\mathcal{T}_\alpha^* \models \mathcal{K}^{\alpha,i} \not\models \phi$. Therefore, a negative counterexample serves only the purpose of updating $end(\alpha)^i$ for some $\alpha \in \mathcal{V}_p^i$, $\alpha < \gamma$. By definition, $\mathsf{val}(\phi, \mathcal{H}^i) = \uparrow_p\left(\frac{\alpha + end(\alpha)^i}{2}\right)$, and $end(\alpha)^{i+1}$ (by definition) will be equal to $\uparrow_p\left(\frac{\alpha+\gamma}{2}\right) - 10^{-p}$. This means that for each $\alpha \in \mathcal{V}_p^i$ this case happens at most $log_2(\alpha, end(\alpha)^i) \leq log_2(0,1]_p \leq log_2 10^p < 4p$ times. We have shown that the number of negative examples is bounded by a polynomial w.r.t. $|\mathcal{T}|$, p, and the size of the largest counterexample seen so far. It follows that N is also bounded by $|\mathcal{T}|$ and the size of the largest counterexample seen so far.

Termination. $L_{\mathfrak{F}_\pi}$ performs at most polynomially many steps with respect to $|\mathcal{T}|$, the size of the largest counterexample received so far, and p until at iteration i, either it knows that $p < \mathsf{prec}(\mathcal{T})$ or it has computed for every $\beta \in \mathcal{T}^v$, $\mathcal{K}^{\alpha,i} \equiv \mathcal{T}_\beta^*$ with $\alpha \in \mathcal{V}_p^i$ and $\beta = mid(\alpha)^i$ (because $\mathfrak{F}$ is safe). Every L_α with $\alpha \in \mathcal{V}_p^i$ satisfies $\alpha \leq \gamma < end(\alpha)$ for $\gamma \in \mathcal{T}^v$ because $(\phi, \uparrow_p(\alpha)) \in \mathsf{P}^i$. Since we have shown that the number of positive counterexamples that L_π receives is polynomially bounded, for every $\gamma \in \mathcal{T}^v$, there is $(\phi, \gamma') \in \mathsf{P}^i$ such that $\mathsf{val}(\phi, \mathcal{T}) = \gamma'$. As described, the learner will be able to identify the correct valuation for every formula $(\phi, \gamma') \in \mathsf{P}^i$ in polynomial time w.r.t. $|\mathcal{T}|$, the largest counterexample seen so far, and p (if the estimated precision is high enough). When this happens, we have that $\mathcal{T}^v \subseteq \mathcal{H}^{i,v}$ and by Lemma 4.9, the learner $L_{\mathfrak{F}_\pi}$ finds a hypothesis $\mathcal{H}^i$ such that $\mathcal{H}^i \equiv \mathcal{T}$. □

By asking only equivalence queries, the learner may be able to find a lower bound for

[3] We can assume that, before building $\mathcal{H}^i$, every L_α receives a positive counterexample ϕ if there is $\phi \in \mathcal{K}^{\beta,i}$ with $\beta \in \mathcal{V}_p^i$ and $\beta > \alpha$ such that $\mathcal{K}^{\alpha,i} \not\models \phi$.

the precision of the target. This is done by keeping track of counterexamples received so far. Example 4.32 gives a simple run of the steps described in the previous lemma about how to make this check.

Example 4.32. Let $\mathfrak{F} = (\mathcal{E}, \mathcal{L}, \mu)$ be a safe learning framework where $\mathcal{L}$ is the set of all possible conjunctions of literals over variables $\mathsf{V} = \{\mathsf{v}_1, \mathsf{v}_2\}$, and $\mathcal{E}$ is the set of literals. Let $\mathfrak{F}_\pi$ be its possibilistic extension, let $t = \{(\mathsf{v}_1, 0.33), (\mathsf{v}_2, 0.7)\}$ be the target and let the estimated precision p of the target be 1. We consider the polynomial time learning algorithm for learning conjunction of literals that starts with the empty hypothesis and when it receives a counterexample, it adds it to the hypothesis. Following our argument in Lemma 4.29, we ask the equivalence query with input $\emptyset$ and assume we receive the positive counterexample $(\mathsf{v}_1, 0.19)$. As explained, the possibilistic learner L will create the hypothesis $\mathcal{H} = \{(\mathsf{v}_1, 0.6)\}$ because $0.6 = \uparrow_p \left(\frac{\uparrow_p(0.19)+1}{2}\right)$ and we will call again the equivalence query oracle with input $\mathcal{H}$. Assuming the call returns the negative counterexample $(\mathsf{v}_1, 0.35)$, L checks if $p < \mathsf{prec}(\mathcal{T})$ according to the counterexamples received. Since the information revealed is not enough to discover that $p < \mathsf{prec}(\mathcal{T})$, L will build the hypothesis $\mathcal{H} = \{(\mathsf{v}_1, 0.2)\}$. After another equivalence query, assume L receives the positive counterexample $(\mathsf{v}_1, 0.23)$. When this happens, L discovers that $p < \mathsf{prec}(\mathcal{T})$ because it knows $\mathcal{T} \models (\mathsf{v}_1, 0.23)$ and $\mathcal{T} \not\models (\mathsf{v}_1, 0.35)$. Assuming $p = 1$, this would imply that $\mathcal{T} \models (\mathsf{v}_1, 0.3)$ and $\mathcal{T} \not\models (\mathsf{v}_1, 0.3)$ because $\uparrow_p(0.23) = 0.3$ and $\downarrow_p(0.32) = 0.3$. Therefore, $\mathsf{prec}(\mathcal{T}) \geq 2$. ◁

Once the precision of the target is correctly estimated, again by following the steps described in Lemma 4.29, in Example 4.33 the learner can use the information of counterexamples received so far to both identify the intervals where the valuations present in the target lie, and estimate the valuation of formulas present in the hypothesis.

Example 4.33. Assume now that the target is $\mathcal{T} = \{(\mathsf{v}_1, 0.3), (\mathsf{v}_2, 0.7)\}$ and $p = 1$. Assume we have $\mathsf{N} = \{(\mathsf{v}_2, 0.9), (\mathsf{v}_2, 1), (\mathsf{v}_1, 0.43)\}$ and $\mathsf{P} = \{(\mathsf{v}_1, 0.13), (\mathsf{v}_2, 0.42)\}$, which are respectively the sets of negative and positive counterexamples received so far. As described in the previous lemma, the learner L computes the hypothesis $\mathcal{K}^{0.2} = \{\mathsf{v}_1 \wedge \mathsf{v}_2\}$, $\mathcal{K}^{0.5} = \{\mathsf{v}_2\}$ and the hypothesis $\mathcal{H} = \{(\mathsf{v}_1, 0.3), (\mathsf{v}_2, 0.7)\}$, which is equivalent to $\mathcal{T}$. The number 0.3 is obtained after computing $\uparrow_p \left(\frac{\uparrow_p(0.13)+\downarrow_p(0.4)}{2}\right)$ because of the positive counterexample $(\mathsf{v}_1, 0.13)$. ◁

The statement in Lemma 4.29 can be used to claim polynomial time transferability results from classical to possibilistic settings.

Theorem 4.34. *Let $\mathfrak{F}$ be a safe FO learning framework in* ELP(EQ) *such that there is a positive bounded learner for $\mathfrak{F}$. Then, $\mathfrak{F}_\pi$ is in* ELP(EQ).

Proof. We consider the learning system $(L_{\mathfrak{F}_\pi}, T_{\mathfrak{F}_\pi}(\mathcal{T}))$ on input $\Sigma_\mathcal{T}$. $L_{\mathfrak{F}_\pi}$ starts by estimating the precision p of the target $\mathcal{T} \in \mathcal{L}_\pi$ to be 1. Using Lemma 4.29, we can assume that this learner can either determine that $p < \mathsf{prec}(\mathcal{T})$ or find a hypothesis $\mathcal{H}$ such that $\mathcal{H} \equiv \mathcal{T}$, in time polynomial w.r.t. $|\mathcal{T}|$, p and the largest counterexample seen so far using only equivalence queries. In the former case, this learner sets the estimated precision p of the target to $p+1$. This happens at most $\mathsf{prec}(\mathcal{T})$ times, which is bounded by $|\mathcal{T}|$. As a consequence, $\mathfrak{F}_\pi$ is in ELP(EQ). □

The other direction also holds. The polynomial time transferability result from the possibilistic to classical case (Theorem 4.35) does not require the assumption that learning frameworks are safe.

Theorem 4.35. *Let $\mathfrak{F}_\pi$ be the possibilistic extension of an FO learning framework $\mathfrak{F}$. If $\mathfrak{F}_\pi$ in* ELP(EQ)*, then $\mathfrak{F}$ is in* ELP(EQ).

Reduction With Membership and Equivalence Queries

A learner that has access to both membership and equivalence query oracle has a way of finding the precision of the target when it is unknown. With membership queries, we can use Algorithm 6 to find the valuation of formulas up to a given precision. By Lemma 4.36, we can obtain useful information about the precision of the target with the counterexamples obtained after an equivalence query.

Lemma 4.36. *Assume $\mathfrak{F}_\pi = (\mathcal{E}_\pi, \mathcal{L}_\pi, \mu_\pi)$ is the possibilistic extension of a safe FO learning framework in* ELP(MQ,EQ) *and $\mathcal{T} \in \mathcal{L}_\pi$ is the target. Given $p \in \mathbb{N}^+$, one can determine that $p < \mathsf{prec}(\mathcal{T})$ or compute $\mathcal{H} \in \mathcal{L}_\pi$ such that $\mathcal{H} \equiv \mathcal{T}$, in polynomial time with respect to $|\mathcal{T}|$, p, and the largest counterexample seen so far.*

Proof. In our proof, we use the following claims.

Claim 4.37. *Given $\mathcal{H} \in \mathcal{L}_\pi$ such that $\mathcal{T} \models \mathcal{H}$, one can construct in polynomial time in $|\mathcal{H}|$ some $\mathcal{H}' \in \mathcal{L}_\pi$ such that $\mathcal{T} \models \mathcal{H}' \models \mathcal{H}$ and, for all $(\phi, \alpha) \in \mathcal{H}'$, $\mathcal{T} \models (\phi, \alpha)$ and $\downarrow_{\mathsf{prec}(\mathcal{H}')}(\alpha) = \mathsf{val}(\phi, \mathcal{T})$.*

Proof. Let $\mathcal{H}'$ be the set of all (ϕ, β) such that $(\phi, \alpha) \in \mathcal{H}$ and Algorithm 6 returns β with ϕ and $\mathsf{prec}(\mathcal{H})$ as input. As $\mathcal{T} \models \mathcal{H}$, by construction of $\mathcal{H}'$, $\mathcal{T} \models \mathcal{H}' \models \mathcal{H}$. By Lemma 4.22, $\mathcal{H}'$ can be constructed in polynomial time in $|\mathcal{H}|$ and is as required. □

Claim 4.38. *Let $\mathcal{H} \in \mathcal{L}_\pi$ be such that, for all $(\phi, \alpha) \in \mathcal{H}$, $\mathcal{T} \models (\phi, \alpha)$ and $\downarrow_{\mathsf{prec}(\mathcal{H})}(\alpha) = \mathsf{val}(\phi, \mathcal{T})$. If $\mathsf{EQ}_{\mathfrak{F}_\pi, \mathcal{T}}$ with input $\mathcal{H}$ returns (ϕ, α) then either $\mathsf{prec}(\mathcal{T}) > \mathsf{prec}(\mathcal{H})$ or $\mathcal{H}^*_\beta \not\models \phi$ where $\downarrow_{\mathsf{prec}(\mathcal{H})}(\beta) = \mathsf{val}(\phi, \mathcal{T})$.*

Proof. By Point 3 of Proposition 4.7, $\mathcal{H}^*_\beta \models \phi$ iff $\mathcal{H} \models (\phi, \beta)$. If $\mathcal{H} \models (\phi, \beta)$ or $\beta = 0$ (note: β can be 0 because, e.g., $\downarrow_1(0.01) = 0$), then $\mathsf{prec}(\mathsf{val}(\phi, \mathcal{T})) > \mathsf{prec}(\mathcal{H})$. By Point 5 of Proposition 4.7, $\mathsf{val}(\phi, \mathcal{T}) \in \mathcal{T}^v \cup \{1\}$, so $\mathsf{prec}(\mathcal{T}) > \mathsf{prec}(\mathcal{H})$. □

By Remark 4.3, we can assume at all times in this proof that any hypothesis constructed is entailed by the target (possibilistic or not). Moreover, by Claim 4.37, we can assume that, for any target and hypothesis $\mathcal{T}, \mathcal{H} \in \mathcal{L}_\pi$, we have that, for all $(\phi, \alpha) \in \mathcal{H}$, $\mathcal{T} \models (\phi, \alpha)$ and $\downarrow_{\mathsf{prec}(\mathcal{H})}(\alpha) = \mathsf{val}(\phi, \mathcal{T})$. So we can assume at all times in our proof that the hypothesis $\mathcal{H}$ we construct (Equation 4.1) satisfies the conditions of Claim 4.38. Since $\mathfrak{F}$ is in EL(MQ,EQ), there is a learning system $(L_\mathfrak{F}, T_\mathfrak{F})$ such that, for any $\mathcal{K} \in \mathcal{L}$, $L_\mathfrak{F}$ calls both $\mathsf{MQ}_{\mathfrak{F},\mathcal{K}}$ and $\mathsf{EQ}_{\mathfrak{F},\mathcal{K}}$, and $\mathcal{K}' \in (L_\mathfrak{F}, T_\mathfrak{F})(\Sigma_k)$ implies $\mathcal{K}' \equiv \mathcal{K}$.

As in the proof of Theorem 4.24, we consider the learning system $(L_{\mathfrak{F}_\pi}, T_{\mathfrak{F}_\pi}(\mathcal{T}))$ on input $(\Sigma_\mathcal{T})$ where $\mathcal{T} \in \mathcal{L}_\pi$ is the target and we describe the steps that $L_{\mathfrak{F}_\pi}$ makes on input $\Sigma_\mathcal{T}$. For some $\alpha \in (0,1]_p$, $L_{\mathfrak{F}_\pi}$ does every step that $L_\mathfrak{F}$ performs in a path p_α in $\mathsf{T}_{L_\mathfrak{F}, T_\mathfrak{F}(\mathcal{T}^*_\alpha), \Sigma_{\mathcal{T}^*}}$ (Remark 4.2). Since $\mathfrak{F}$ is safe, for every $\beta \in (0,1]$, $\mathcal{T}^*_\beta \in \mathcal{L}$. We denote by R the set of valuations α that $L_{\mathfrak{F}_\pi}$ is considering in order to learn $\mathcal{K}^\alpha$ equivalent to $\mathcal{T}^*_\alpha$. We write $\mathcal{K}^{\beta,n}$ to indicate that the learner $L_{\mathfrak{F}_\pi}$ has reached n configurations in p_β that correspond to when the learner stops in the equivalence query state with input $\mathcal{K}^{n,\beta}$. For $n = 0$, we assume that $\mathcal{K}^{\beta,n} = \emptyset$.

Description of $L_{\mathfrak{F}_\pi}$'s steps. Initially, $\mathsf{R} := \{10^{-p}\}$. Whenever for $\beta \in \mathsf{R}$ in p_β there is a call to $\mathsf{EQ}_{\mathfrak{F},\mathcal{T}^*_\beta}$ with input $\phi \in \mathcal{E}$, by Point 3 of Proposition 4.7, $L_{\mathfrak{F}_\pi}$ simulates $\mathsf{MQ}_{\mathfrak{F},\mathcal{T}^*_\beta}$ by calling $\mathsf{MQ}_{\mathfrak{F}_\pi,\mathcal{T}}$ with (ϕ, β) as input and treating its answer as the answer of $\mathsf{EQ}_{\mathfrak{F},\mathcal{T}^*_\beta}$. Let $\mathcal{H}_0$ be $\{(\phi_\top, \alpha)\}$ where $\phi_\top$ is a tautology and α is a valuation with $\mathsf{prec}(\alpha) = p$. When $L_{\mathfrak{F}_\pi}$ reaches the configuration in all p_α, $\alpha \in \mathsf{R}$ corresponding to an equivalence query state with input $\mathcal{K}^{\alpha,m_\alpha}$, $L_{\mathfrak{F}_\pi}$ creates

$$\mathcal{H} := \bigcup_{\alpha \in \mathsf{R}} \{(\phi, \alpha) \mid \phi \in \mathcal{K}^{\alpha,m}\} \cup \mathcal{H}_0 \tag{4.1}$$

and calls $\mathsf{EQ}_{\mathfrak{F}_\pi,\mathcal{T}}$ with $\mathcal{H}$ as input. If the answer is 'yes', $L_{\mathfrak{F}_\pi}$ has computed $\mathcal{H}$ such that $\mathcal{H} \equiv \mathcal{T}$ and it terminates. Upon receiving a (positive) counterexample (ϕ, γ), $L_{\mathfrak{F}_\pi}$ runs Algorithm 6 with ϕ and $\mathsf{prec}(\mathcal{H})$ as input and it computes a valuation β such that $\downarrow_{\mathsf{prec}(\mathcal{H})}(\beta) = \mathsf{val}(\phi, \mathcal{T})$ (Lemma 4.22). If $\beta \notin \mathsf{R}$, $L_{\mathfrak{F}_\pi}$ starts the execution of every step made by $L_\mathfrak{F}$ in a path p_β in $\mathsf{T}_{L_\mathfrak{F}, T_\mathfrak{F}(\mathcal{T}^*_\beta), \Sigma_{\mathcal{T}^*}}$ and it adds β to R. Otherwise, $\beta \in \mathsf{R}$ and $L_{\mathfrak{F}_\pi}$ checks whether $\mathcal{K}^{\beta,m} \models \phi$ (Remark 4.4) (assume m is the number of equivalence queries posed so far in the visited part of the path p_β). If $\mathcal{K}^{\beta,m} \models \phi$ then, by Claim 4.38, $L_{\mathfrak{F}_\pi}$ knows that $\mathsf{prec}(\mathcal{H}) < \mathsf{prec}(\mathcal{T})$ then it terminates. If $\mathcal{K}^{\beta,m} \not\models \phi$ (Remark 4.4) then ϕ is a (positive) counterexample for $\mathcal{K}^{\beta,m}$ and $\mathcal{T}^*_\beta$. $L_{\mathfrak{F}_\pi}$ treats ϕ as a counterexample returned

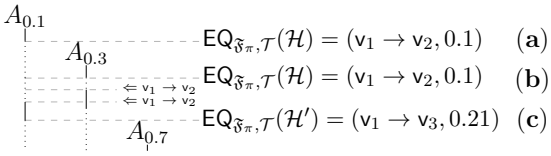

Figure 4.2: Multiple instances of algorithm A in Example 4.39. Time flows top-down. A dotted line means that the learner is waiting in query state, a continuous line means that the learner is running.

by $\mathsf{EQ}_{\mathfrak{F},\mathcal{T}^*_\alpha}$ for every $\alpha \in \mathsf{R}$ such that $\alpha \leq \beta$ and $\mathcal{K}^{\alpha,m} \not\models \phi$. Observe that, since $\mathcal{H}_0 \subseteq \mathcal{H}$, by the construction of $\mathcal{H}$, at all times $\mathsf{prec}(\mathcal{H}) = p$.

Termination. We now argue that $L_{\mathfrak{F}_\pi}$ terminates in polynomially many steps w.r.t. $|\mathcal{T}|$, p, and the largest counterexample seen so far. Since there is only one instance β in R for each valuation β such that $\downarrow_p(\beta) = \mathsf{val}(\phi, \mathcal{T})$, by Point 5 of Proposition 4.7, we have that at all times $|\mathsf{R}|$ is linear in $|\mathcal{T}^v|$, which is bounded by $|\mathcal{T}|$. By Lemma 4.22, whenever $L_{\mathfrak{F}_\pi}$ runs Algorithm 6 to compute a valuation with ϕ and p as input, only polynomially many steps in $|\phi|$ and p are needed. Since $\mathfrak{F}$ is safe and L is a polynomial time learner for $\mathfrak{F}$ either we can determine that $p < \mathsf{prec}(\mathcal{T})$ or $L_{\mathfrak{F}_\pi}$ computes in polynomial time in the size of $\mathcal{T}^*_\beta$ and the largest counterexample seen so far, and outputs $\mathcal{K}^{\beta,n} = \mathcal{H}^*_\beta$ such that $\mathcal{H}^*_\beta \equiv \mathcal{T}^*_\beta$. In this case, by Lemma 4.9, $\mathcal{H} \equiv \mathcal{T}$ and $L_{\mathfrak{F}_\pi}$ terminates. □

The constructive proof of Lemma 4.36 delineates the steps made in Example 4.39 where the precision of the target is 1.

Example 4.39. Let $\mathfrak{F} = (\mathcal{E}, \mathcal{L}, \mu)$ be the safe learning framework where $\mathcal{L}$ is the set of all propositional Horn KBs, $\mathcal{E}$ is the set of all (propositional) Horn clauses and μ maps Horn KBs to Horn clauses that are entailed by it. Let $\mathcal{T} \in \mathcal{L}_\pi$ and L be, respectively, the target and the learner of Example 4.26. Following our argument in Lemma 4.36, we start performing the steps that a classical learner L for Horn KBs may do to learn $\mathcal{T}^*_{0.1}$. We add 0.1 to R. When we should wait in equivalence query state, we build $\mathcal{H} = \{(\phi_\top, 0.1)\}$ (Equation A.1) and call $\mathsf{EQ}_{\mathfrak{F}_\pi,\mathcal{T}}$ with $\mathcal{H}$ as input (Point (a) in Figure 4.2). Assume we receive the positive counterexample $(\mathsf{v}_1 \to \mathsf{v}_2, 0.1)$. We run Algorithm 6 with 1 and $(\mathsf{v}_1 \to \mathsf{v}_2)$ as input, which computes $\mathsf{val}(\mathsf{v}_1 \to \mathsf{v}_2, \mathcal{T}) = 0.3$. Since $0.3 \notin \mathsf{R}$, we start performing the steps that a classical learner L for Horn KBs may do to learn $\mathcal{T}^*_{0.3}$. When we reach a point where all learners are waiting in the query state, we call again $\mathsf{EQ}_{\mathfrak{F}_\pi,\mathcal{T}}$ with $\mathcal{H}$ as input (Point (b) in Figure 4.2).

Assume we receive $(\mathsf{v}_1 \to \mathsf{v}_2, 0.1)$ again. We have that $\mathsf{val}(\mathsf{v} \to \mathsf{v}_2, \mathcal{T}) = 0.3$ and $0.3 \in \mathsf{R}$. Since $\mathcal{K}^{0.3,1} \not\models \mathsf{v}_1 \to \mathsf{v}_2$ and $\mathcal{K}^{0.1,1} \not\models \mathsf{v}_1 \to \mathsf{v}_2$, we treat $\mathsf{v}_1 \to \mathsf{v}_2$ as a positive counterexample for both $\mathcal{K}^{0.3,1}$ and $\mathcal{K}^{0.1,1}$. When we all learners are waiting

in equivalence query state again, we call the oracle $\mathsf{EQ}_{\mathfrak{F}_\pi,\mathcal{T}}$ with input the hypothesis $\mathcal{H}' = \{(\phi_\top, 0.1), (\mathsf{v}_1 \rightarrow \mathsf{v}_2, 0.1), (\mathsf{v}_1 \rightarrow \mathsf{v}_2, 0.3)\}$.

Assume the response is $(\mathsf{v}_1 \rightarrow \mathsf{v}_3, 0.21)$. We run Algorithm 6 with arguments 1 and $(\mathsf{v}_1 \rightarrow \mathsf{v}_3)$ as input, which returns $\mathsf{val}(\mathsf{v}_1 \rightarrow \mathsf{v}_3, \mathcal{T}) = 0.7$. As before, we compute every step that a classical learner L for Horn KBs may do to learn $\mathcal{T}^*_{0.7}$ (Point (c) in Figure 4.2) and add 0.7 to R. When all learners are waiting again we call $\mathsf{EQ}_{\mathfrak{F}_\pi,\mathcal{T}}$ with $\mathcal{H}'$ as input. Assume we receive $(\mathsf{v}_1 \rightarrow \mathsf{v}_3, 0.1)$. We then send $\mathsf{v}_1 \rightarrow \mathsf{v}_3$ to every learner in R. Next time we call $\mathsf{EQ}_{\mathfrak{F}_\pi,\mathcal{T}}$, with $\mathcal{H}' \cup \{(\mathsf{v}_1 \rightarrow \mathsf{v}_3, 0.7), (\mathsf{v}_1 \rightarrow \mathsf{v}_3, 0.3), (\mathsf{v}_1 \rightarrow \mathsf{v}_3, 0.1)\}$ as input. The answer is 'yes' and we are done. ◁

In some cases, the learner can discover if the precision of the hypothesis needs to increase. Example 4.40 shows such case.

Example 4.40. Assume the target is $\mathcal{T} \coloneqq \{(\mathsf{v}_1 \rightarrow \mathsf{v}_2, 0.32)\}$ and the learner built the hypothesis $\mathcal{H} = \{(\mathsf{v}_1 \rightarrow \mathsf{v}_2, 0.3)\}$. Similarly to Example 4.39, the precision of the hypothesis is set to 1. A future equivalence query will return the counterexample $\mathcal{H} \coloneqq \{(\mathsf{v}_1 \rightarrow \mathsf{v}_2, \alpha)\}$ with $\alpha > 0.3$. The learner will run Algorithm 6 with input 1 and $\mathsf{v}_1 \rightarrow \mathsf{v}_2$, which will return 0.3. Since $\mathcal{H} \models (\mathsf{v}_1 \rightarrow \mathsf{v}_2, 0.3)$, this can happen only if the precision of the hypothesis is low. ◁

A direct consequence of Lemma 4.36 is Theorem 4.41.

Theorem 4.41. *Let $\mathfrak{F}$ be a safe FO learning framework. We have that $\mathfrak{F}$ is in* ELP(MQ,EQ) *iff $\mathfrak{F}_\pi$ is in* ELP(MQ,EQ)*.*

Proof. One direction holds by Theorem 4.19. We prove the other direction. Let $\mathfrak{F}$ be a safe FO learning framework in ELP(MQ,EQ) and let $\mathfrak{F}_\pi = (\mathcal{E}_\pi, \mathcal{L}_\pi, \mu_\pi)$ be its possibilistic extension. Consider a learner that initially estimates the precision p of the target $\mathcal{T} \in \mathcal{L}_\pi$ to be 1. By Lemma 4.36, we can assume that this learner can either determine that $p < \mathsf{prec}(\mathcal{T})$ or find a hypothesis $\mathcal{H}$ such that $\mathcal{H} \equiv \mathcal{T}$, in polynomial time w.r.t. $|\mathcal{T}|$, p and the largest counterexample seen so far. In the former case, this learner sets the estimated precision p of the target to $p+1$. This happens at most $\mathsf{prec}(\mathcal{T})$ times, which is bounded by $|\mathcal{T}|$. As a consequence, $\mathfrak{F}_\pi$ is in ELP(MQ,EQ). □

Reduction With Subset and Superset Queries

We know by Theorem 4.15 that without knowing the precision of the target, it is not possible to transfer polynomial time learnability results with only subset or superset

queries. In this section, we provide polynomial time reduction in both directions (classical to possibilistic and vice-versa) for safe learning frameworks when the learner can ask only subset or superset queries and the precision of the target is fixed. We recall the need of assuming that the learning framework under consideration is safe when showing the direction from classical to possibilistic setting (Example 4.20).

Theorem 4.42. *Let $\mathfrak{F}$ be a safe FO learning framework. For all $p \in \mathbb{N}^+$, $\mathfrak{F}$ is in* ELP(SbQ) *iff $\mathfrak{F}_\pi^p$ is in* ELP(SbQ) *and $\mathfrak{F}$ is in* ELP(SpQ) *iff $\mathfrak{F}_\pi^p$ is in* ELP(SpQ).

We can simulate an equivalence query with input a hypothesis $\mathcal{H}$ by first asking a superset and then a subset query with input $\mathcal{H}$. A membership query with input a formula ϕ is simulated by a subset query with input $\{\phi\}$. If both superset and subset queries are allowed, we can use the result of Theorem 4.41 to state the next corollary.

Corollary 4.43. *Let $\mathfrak{F}$ be a safe FO learning framework. If $\mathfrak{F}$ is in* ELP(MQ,EQ)*, then $\mathfrak{F}_\pi$ is in* ELP(SbQ,SpQ).

Connections With the PAC Learning Model

We end this section recalling a connection between the exact and the PAC learning models. In the PAC model [Valiant, 1984], a learner receives classified examples and attempts to create a hypothesis that approximates the target. In this model, it is assumed the existence of the example oracle EX that when called, it draws an example according to a probability distribution and it returns to the caller the example and the classification of the example according to the target. It is known that polynomial time results for the exact learning model with only equivalence queries can be transferred to the PAC learning model [Angluin, 1988; Mohri et al., 2012]. This holds also in the case where the learner can pose membership and equivalence queries and the PAC model is extended with membership queries—a variant of PAC in which the learner can ask membership queries and example queries (but not equivalence queries). Let PLP(EX) and PLP(EX,MQ) be, respectively, the class of all learning frameworks that are PAC learnable with only example queries in polynomial time and PAC learnable with both example and membership queries in polynomial time. In symbols, ELP(EQ) $\subseteq$ PLP(EX) and ELP(MQ,EQ) $\subseteq$ PLP(EX,MQ). Recall that by Remark 4.4, when a framework is exactly learnable in polynomial time, it is assumed that the logic used to define the hypothesis space allows to check if an example is entailed by it in polynomial time. Then, by Theorem 4.41, the following corollary holds.

Corollary 4.44. *For all safe FO learning frameworks $\mathfrak{F}$, if $\mathfrak{F} \in$* ELP(MQ,EQ)*, then $\mathfrak{F}_\pi \in$* PLP(EX,MQ).

In general, the converse direction of Corollary 4.44 does not hold as shown by Blum [1994] with an argument based on the assumption that one-way functions exist and by Ozaki et al. [2020] relying on a representation-dependent proof. Moreover, by Theorem 4.34 the next corollary is also valid.

Corollary 4.45. *For all safe FO learning frameworks $\mathfrak{F}$ such that there is a polynomial time and positive bounded learner for $\mathfrak{F}$, if $\mathfrak{F} \in \text{ELP}(\mathsf{EQ})$, then $\mathfrak{F}_\pi \in \text{PLP}(\mathsf{EX})$.*

4.4 Discussion

Our results focus on the learning from entailments setting, where prior to learning, we select a logic language that describes concepts (logic theories). Information about an unknown target concept is given by examples that are logic formulas, and the target is identified by looking at the entailment relation between the examples and the target. The goal is to find, through the information provided by the examples, a hypothesis that is logically equivalent to the target. There are other settings for learning logic theories such as learning from satisfiability, where examples are set of formulas and classified as positive example if when conjoined with the unknown target concept the resulting formula is satisfiable. Otherwise, they are negative examples. Some relationships can already be found in literature. Hermo and Ozaki (2020) show (in Section $C.1$) how to transfer polynomial time learnability results from entailments to interpretations when the logic language considered is expressive up to multivalued dependency formulas. Raedt (1997) also shows that learning (not polynomial time learning) from interpretations reduces to learning from entailments. In turn, learning from entailments reduces to learning from satisfiability. Therefore, some of the learnability results presented also hold in the learning from partial interpretations and learning from satisfiability settings.

Future works can investigate learnability and polynomial time transferability results to possibilistic settings in learning from interpretations, or learning from satisfiability. We already know that learnability from partial interpretations reduces to learnability from entailment and vice-versa, but whether polynomial time transferability results hold in possibilistic settings is an open question. We can also consider studying learnability and polynomial time transferability results for the PAC learning model that is more general and includes strictly more polynomial time learnable frameworks than the exact model.

Chapter 5

Exact Learning Possibilistic Horn Theories

The theoretical framework and the results shown in the previous chapter inspired the development of a learning algorithm that identifies unknown target possibilistic Horn formulas by asking queries as defined in the *Exact Learning* (EL) model. This chapter is divided in two parts. In the first, the goal is to identify an efficient method with which we can use exact learning algorithms in practice. Indeed, there are cases in which no oracle able to answer queries is available, especially equivalence queries. In principle, domain experts could be available to answer queries to exact learning algorithms, but it is a costly and hard task to complete. We propose a semi-automatic technique that can simulate such oracles. In the second part, we tackle the problem of learning possibilistic logic Horn formulas. Since this study is based on the EL model, we are able to prove guarantees on the learning outcome. Moreover, this work gives the opportunity to express the uncertainty concerning the learned hypothesis of machine learning models.

For simulating oracles, we propose a solution based on the connection between the EL and the PAC model (end of Section 4.3 in Chapter 4). The idea is that instead of aiming at the exact identification of the target concept, we can guarantee PAC learnability conditions if counterexamples for equivalence queries are found in a randomly generated sample of labelled examples. We test this idea with a modification of the LRN algorithm [Frazier and Pitt, 1993b] that learns (propositional) Horn formulas in the EL model by asking membership and equivalence queries from partial interpretations. We name our version of the algorithm HORN^* and we show that it runs in polynomial time with respect to the size of the target formula and the number of variables in the language. Since Horn is a logic upon which automated reasoning can be carried on efficiently [Dowling and Gallier, 1984], one can quickly check if a property under the form of a rule holds, or if

an unintended rule is entailed.

Tabular datasets, that collect examples in the form of features of vector and target value, are abundant and easily accessible. Our method proposes to train a machine learning model from data and use such models to simulate oracles. Employing trained models to automate the task of oracles has two main advantages. First, we can use the generalisation capabilities of machine learning models to obtain comprehensively answers that cannot be found in the dataset. Secondly, if the machine learning model used to answer queries is a so called black-box (like ANNs), we can use an exact learning algorithm to extract rules encoded in it. *Artificial Neural Network* (ANN) are being widely adopted for giving suggestions that may impact the life course of a person (criminal justice, health, and so on [McGough, 2018; Wexler, 2017]). Therefore, we carried our tests treating a trained neural network model N as the target. More in detail, our method treats an ANN N as an oracle that encodes a Horn formula, and uses the HORN^* algorithm to ask queries to N and gradually form a hypothesis that approximates the Horn rules encoded in N. It is often the case that not all values in a dataset are known or trustable. For this reason, our approach assumes settings in which the dataset used to train the neural network contains noisy or missing values. Moreover, there is no assumption regarding the internal architecture of the neural network. In this way, when the network is trained on medical data, we are able to extract rules of the form 'if an adult is not a smoker and has normal pressure, then it survives the treatment', in Horn logic:

$$((\mathsf{adult} \wedge \neg \mathsf{smoker} \wedge \mathsf{normal\_pressure}) \rightarrow \mathsf{survives}).$$

We perform an empirical study using the hepatocellular carcinoma dataset (HCC) [Santos et al., 2015], which describes survivability of patients diagnosed with hepatocellular carcinoma according to clinical information. The HCC dataset contains many missing values of attributes of patients. We compare the hypothesis built with our approach with the hypothesis built by a state-of-the-art implementation of the incremental decision tree algorithm [Domingos and Hulten, 2000b]. Our rule extraction procedure correctly extracts meaningful rules and it is two times faster than the decision tree algorithm.

In the second part of this chapter, we develop and test the Π_HORN^* algorithm that exactly identifies an unknown possibilistic (propositional) Horn formula by asking equivalence queries and possibility queries. The latter is similar to a membership query, but it takes into account possibility degrees. We provide theoretical results concerning polynomial time learnability of possibilistic logic theories in the exact learning model. Π_HORN^* runs in polynomial time with respect to the size of the target and the number of variables in the language. Also in this case, we test the strategy for simulating oracles

with a trained neural network. The additional advantage of using Π_HORN* is that with possibilistic rules we can represent the uncertainty of rules encoded in the trained ANN. Also in the second part, we consider the case where a neural network N has been trained on uncertain or unreliable data, and compare the hypothesis built by Π_HORN* (that treats N as an oracle) with the hypothesis built by a decision tree algorithm [Domingos and Hulten, 2000a; Montiel et al., 2018].

In Section 5.1, we present HORN*, that identifies a Horn theory in the learning from partial interpretation setting, and we explain how to simulate oracles. We provide experimental results in Section 5.2. Then, in Section 5.3, we explain how to identify possibilistic Horn theories with Π_HORN*, and we show the results of extracting possibilistic Horn rules from trained ANN models. The implementation of both algorithm is available at the following link: `https://git.app.uib.no/Cosimo.Persia/horn`.

5.1 Extracting Horn Rules

We present an adaptation of the LRN algorithm [Frazier and Pitt, 1993b] that learns from partial interpretations instead of entailments, as originally proposed by the authors of the mentioned paper. This algorithm is able to exactly identify any unknown target Horn formula by posing queries to membership and equivalence query oracles. The algorithm is guaranteed to terminate in polynomial time with respect to the number of variables into consideration. After that, we show how to simulate membership and equivalence oracles with machine learning models trained on data.

The HORN* Algorithm

Angluin et al. (1992) proposed for the first time an algorithm for learning conjunction of Horn clauses from classical (not partial) interpretations that runs in polynomial time with respect to the size of the target, and the number of variables into consideration. Later, Frazier and Pitt (1993a) showed that conjunctions of Horn clauses can be learned also in the learning from entailment setting with their LRN algorithm. Exact identification of a Horn theory with LRN is achieved in polynomial time with respect to the size of the target Horn formula and the number of variables into consideration. The LRN algorithm is more fit in cases where we cannot assume to have complete information over the training examples [De Raedt, 1997]. As opposed to learning from interpretations where an example provides the truth value of every variable, in learning from entailment the learner has to identify the target in presence of incomplete information. Example 5.1

clarifies this statement.

Example 5.1. *Let $\mathfrak{F}$ be the learning framework $(\mathcal{E}, \mathcal{L}, \mu)$ with $\mathcal{E}$ and $\mathcal{L}$ the set of all Horn formulas of 4 variables ($|\mathsf{V}| = 4$), and μ the logical entailment operator between formulas. Let*

$$\mathcal{T} \coloneqq \{(\mathsf{v}_1 \to \mathsf{v}_2), (\mathsf{v}_1 \to \mathsf{v}_3)\} \in \mathcal{L}$$

be the target. If the learner asks an equivalence query $\mathsf{EQ}_{\mathfrak{F},\mathcal{T}}$ with input $\emptyset$ it may receive the counterexample $(\mathsf{v}_1 \to \mathsf{v}_2)$. So, it will get the information a constraint in the target is violated when v_1 is set to true and v_2 to false. But, the learner cannot state anything about the truth value of v_3. On the contrary, if $\mathcal{E}$ is the set of interpretations of 4 variables and μ is the satisfiability relation between interpretations and formulas, the learner may receive $\mathcal{I} \coloneqq \{(\mathsf{v}_1, \top), (\mathsf{v}_2, \bot), (\mathsf{v}_3, \bot), (\mathsf{v}_4, \top)\}$ as a counterexample after a call to $\mathsf{EQ}_{\mathfrak{F},\mathcal{T}}$ with input $\emptyset$. In this case, there cannot be uncertainty about the truth value of v_3.

Theorem 5.2 shows a connection between the learning from entailment and learning from partial interpretations setting. With our terminology, this means that an algorithm developed in the learning from entailment setting can be used to identify the target concept in the learning from partial interpretations. Unfortunately, this results does not take into account polynomial time property of EL algorithms.

Theorem 5.2 ([De Raedt, 1997] Theorem 16)**.** *Learning from finite partial interpretations reduces to learning from entailment, and vice versa, learning from entailment reduces to learning from partial interpretations*

In this chapter, we focus our attention on learning Horn formulas from partial interpretations. This setting captures best the type of available information we would like to use in this work when training machine learning models. Indeed, it is common to have available tabular data where each row is an example and columns are features of each example. Since non-binary features (of nominal or continuous type) can be binarised into a set of variables [García et al., 2013], we can treat each row in our dataset as a partial interpretation. Discretization techniques [Peng et al., 2009] may lose important information during the process of deciding which bin to associate to continuous variables, but it is required for applying rule induction algorithms that learn classical logic formulas. Example 5.3 shows how a simple dataset can be binarised.

Example 5.3. *Table 5.1 depicts a dataset with two columns, each one of them is a different feature. Feature F_1 is a continuous variable with possible values in range $[0, 100]$ and feature F_2 is nominal with possible values either 1, 2, or 3.*

We can divide the interval of the feature F_1 in 2 parts: feature $F_{1,1}$ associated to the interval $[0, 50)$, and feature $F_{1,2}$ associated to $[50, 100]$. If the original value of an example

F_1	F_2
27	1
56	2
81	3

Table 5.1: A dataset of 3 examples and 2 features.

$F_{1,1}$	$F_{1,2}$	$F_{2,1}$	$F_{2,2}$	$F_{2,3}$
1	0	1	0	0
0	1	0	1	0
0	1	0	0	1

Table 5.2: The binarised dataset

belongs to the i-th interval, the respective value of the $F_{1,i}$ *feature is set to* 1*, and* 0 *otherwise (first 2 columns in Table 5.2). If the variables to binarise are nominal, the number of features that replace the original is less arbitrary and it coincides with the possible number of values that the nominal feature can have. Since in this case* F_2 *can have* 3 *different values,* F_2 *is replaced with* 3 *other columns. For any example, the value of* $F_{2,i}$ *is set to* 1 *if the respective instance of attribute* F_2 *coincides with the i-th value of* F_2 *(for an arbitrary ordering of values of* F_2*). The last 3 columns in Table 5.2 replaced* F_2 *in Table 5.1 where the arbitrary order on the values is one of the natural numbers.* ◁

We adapt the algorithm for learning conjunction of Horn clauses introduced by Frazier and Pitt (1993b) so that it is able to learn rules from partial interpretations. This algorithm required a membership and an equivalence oracle to exactly identify the target Horn formula. In our setting, membership queries take as input partial interpretations and equivalence queries output counterexamples that are also partial interpretations. Algorithm 7 shows the main steps of the modified algorithm.

HORN* starts with an empty hypothesis and throughout the run, it satisfies the condition $\mathcal{T} \models \mathcal{H}$. Lemma 5.4 is an indirect consequence of having a learner that can ask membership queries (Remark 4.3).

Lemma 5.4 (Frazier and Pitt [1993a] Lemma 4 Adaptation)**.** *Let* $\mathcal{T}$ *be the target Horn formula and* $\mathcal{H}$ *be the hypothesis built by* HORN**. At every step, it holds that* $\mathcal{T} \models \mathcal{H}$*.*

HORN* poses equivalence queries until it receives 'yes' as an answer. Upon receiving a counterexample $\mathcal{I}$, by Lemma 5.4 we know $\mathcal{I}$ falsifies a rule in $\mathcal{T}$ not present in $\mathcal{H}$, because it is a negative counterexample. So, we know that some variables set to true by the received counterexample satisfy the antecedent of a rule and some variables set to false falsify their heads. Therefore, positive variables in counterexamples will denote the

Algorithm 7: HORN*

Input: It is assumed that the learner knows $\mathfrak{F}$ (that is, it knows that the hypothesis should be a Horn theory) but not the target $\mathcal{T}$.
Output: $\mathcal{H}$ such that $\mathcal{H} \equiv \mathcal{T}$.
Let S be the empty sequence.
Denote with $\mathcal{A}_i$ the i-th element of S.
Let $\mathcal{H}$ be the empty hypothesis.
while $\mathsf{EQ}_{\mathfrak{F},\mathcal{T}}(\mathcal{H})$ returns a counterexample $\mathcal{I}$ **do**
  $\hat{\mathcal{A}} \leftarrow \{\mathsf{v} \in \mathsf{V} \mid (\mathcal{H} \wedge \{\mathsf{v}' \in \mathsf{V} \mid \mathcal{I}(\mathsf{v}') = \top\}) \models \mathsf{v}\}$
  if there is $\mathcal{A}_i \in S$ such that $\mathcal{A}_i \cap \hat{\mathcal{A}} \subset \mathcal{A}_i$ and $\mathsf{CON}(\mathcal{A}_i \cap \hat{\mathcal{A}}) \neq \emptyset$ **then**
    replace the first such $\mathcal{A}_i$ with $\mathcal{A}_i \cap \hat{\mathcal{A}}$ in S
  else
    append $\hat{\mathcal{A}}$ to S
  end if
  $\mathcal{H} \leftarrow \emptyset$
  for $\mathcal{I} \in S$ **do**
    $\mathcal{A} \leftarrow \{\mathsf{v} \in \mathsf{V} \mid \mathcal{I}(\mathsf{v}) = \top\}$
    for $\mathsf{u} \in \mathsf{CON}(\mathsf{V}, \mathcal{A})$ **do**
      add the rule $(\bigwedge_{\mathsf{v} \in \mathcal{A}} \mathsf{v}) \rightarrow \mathsf{u}$ to $\mathcal{H}$
    end for
  end for
end while
return $\mathcal{H}$

Algorithm 8: CON

Input: variables $\mathcal{A} \subseteq \mathsf{V}$,
Output: A subset $\mathcal{B}$ of $\mathsf{V} \cup \{\bot\}$ such that $\mathcal{T} \models \mathcal{A} \rightarrow \mathsf{v}$ for all $\mathsf{v} \in \mathcal{B}$
$\mathcal{B} \leftarrow \emptyset$
for $\{\mathsf{u} \mid \mathsf{u} \in \mathsf{V} \cup \{\emptyset\} \setminus \mathcal{A}\}$ **do**
  Let $\mathcal{I}$ be such that for $\mathsf{v} \in \mathsf{V}$, $\mathcal{I}(\mathsf{v}) \leftarrow \begin{cases} 1 & \mathsf{v} \in \mathcal{A} \\ 0 & \mathsf{v} = \mathsf{u} \\ ? & \text{otherwise} \end{cases}$
  if $\mathsf{MQ}_{\mathfrak{F},\mathcal{T}}(\mathcal{I}) =$ 'no' **then**
    add u to $\mathcal{B}$
  end if
end for
return $\mathcal{B}$

antecedent of a rule in $\mathcal{T}$. These set of variables are stored in the list S and they are used to form the antecedents of rules in $\mathcal{T}$. With Algorithm 8, at every loop, the HORN^* algorithm will check for every $\mathcal{A} \in S$,[1] if $\mathcal{T} \models \mathcal{A} \rightarrow \mathsf{v}$ for any $\mathsf{v} \in \mathsf{V}$. This guarantees that the target entails every rule added to the hypothesis. Then, HORN^* tries to refine the last received counterexample with the previous. From Line 7 to 12 in Algorithm 7, HORN^* checks if the counterexample $\mathcal{I}$ is returned due to an antecedent stored in S that is too specific, that is it has too many variables. This is done by checking if the target $\mathcal{T}$ entails a rule r where $\mathsf{ant}(\mathsf{r})$ is a subset of variables in the intersection $\hat{\mathcal{A}}$ between variables $\mathcal{A} \in S$ and variables that must be necessarily set to $\top$ by $\mathcal{I}$ with respect to $\mathcal{H}$. By calling CON with input a set of antecedents, HORN^* checks if the refined set of variables still appears in the antecedent of a rule in $\mathcal{T}$. If it does, then $\mathcal{A} \in S$ is replaced with $\hat{\mathcal{A}}$ that represents a better approximation of the antecedent of a rule in $\mathcal{T}$. Otherwise, $\mathcal{I}$ reveals an antecedent of a new rule entailed by $\mathcal{T}$, and the corresponding positive literals satisfied by $\mathcal{I}$ are directly added to S.

The main loop terminates with HORN^* generating a set of Horn rules where elements (set of variables) in S represent the antecedents of rules in $\mathcal{T}$. Algorithm 8 is used to check what variables should appear as consequents for each antecedent $\mathcal{A} \in S$. By Lemma 5.5, we can formally state that since each element in S denotes a rule antecedent in $\mathcal{T}$ (with no duplicates), the size of S is guaranteed to be polynomial in the size of $\mathcal{T}$.

Lemma 5.5 (Frazier and Pitt [1993a] Lemma 9 Adaptation)**.** *Let $\mathcal{T}$ be the target Horn formula. At any time during a run of HORN^*, for any $\mathcal{A}_1, \mathcal{A}_2 \in S$, there are two distinct Horn rules $\mathsf{r}_1, \mathsf{r}_2 \in \mathcal{T}$ such that $\mathsf{ant}(\mathsf{r}_1) \subseteq \mathcal{A}_1$ and $\mathsf{ant}(\mathsf{r}_2) \subseteq \mathcal{A}_2$.*

Each counterexample will either reveal a superset of variables in an antecedent of a rule in $\mathcal{T}$ or it will refine an already discovered antecedent of a rule. The first case can happen $|\mathcal{T}|$ times and the second at most $|\mathsf{V}|$ times. Based on this intuition, Theorem 5.6 finally states the correctness of the HORN^* algorithm.

Theorem 5.6 (Frazier and Pitt [1993a] Theorem 10 Adaptation)**.** *Let V be a finite set of propositional variables and let $\mathcal{T}$ be the unknown target Horn formula. HORN^* runs in polynomial time with respect to $|\mathcal{T}|$, and $|\mathsf{V}|$, and outputs a hypothesis $\mathcal{H}$ such that $\mathcal{H} \equiv \mathcal{T}$ by asking membership and equivalence queries.*

Therefore, we can state that the learning framework under consideration is efficiently exact learnable.

Corollary 5.7. *Let $\mathfrak{F} \coloneqq (\mathcal{E}, \mathcal{L}, \mu)$ be the learning framework with $\mathcal{E}$ the set of partial interpretation, and $\mathcal{L}$ the Horn logic language over V, and for all $\mathcal{H} \in \mathcal{L}$, $\mu(\mathcal{H}) \coloneqq \{\mathcal{I} \in \mathcal{E} \mid \mathcal{I} \models \mathcal{L}\}$. $\mathfrak{F}$ is in* $\textsc{El}(\mathsf{MQ},\mathsf{EQ})$.

[1] Recall that a set of variables is treated interchangeably as a conjunction.

For clarity, Example 5.8 shows how Algorithm 7 generates a hypothesis by asking queries.

Example 5.8. *Let the target Horn theory* $\mathcal{T}$ *be* $\{\mathsf{v}_1 \wedge \mathsf{v}_2 \rightarrow \mathsf{v}_3\}$ *and let* $\mathsf{V} = \{\mathsf{v}_1, \cdots, \mathsf{v}_5\}$. *The* HORN* *algorithm will start with the empty hypothesis* $\mathcal{H} = \emptyset$. *Upon the first call to the equivalence query with input* $\mathcal{H}$, *assume it receives the counterexample*

$$\mathcal{I}_1 = \{(\mathsf{v}_1, \top), (\mathsf{v}_2, \top), (\mathsf{v}_3, \bot), (\mathsf{v}_4, \bot), (\mathsf{v}_5, \top)\}.$$

Since at the first iteration S *is equal to the empty set,* HORN* *will add the set* $\{\mathsf{v}_1, \mathsf{v}_2, \mathsf{v}_5\}$ *to* S *in Line 11. Then, it will generate the hypothesis* $\mathcal{H} = \{(\mathsf{v}_1 \wedge \mathsf{v}_2 \wedge \mathsf{v}_5 \rightarrow \mathsf{v}_3)\}$ *and it will call again the equivalence query oracle with input the new* $\mathcal{H}$. *The only counterexample that can be returned at this stage is*

$$\mathcal{I}_2 = \{(\mathsf{v}_1, \top), (\mathsf{v}_2, \top), (\mathsf{v}_3, \bot), (\mathsf{v}_4, \bot), (\mathsf{v}_5, \bot)\}.$$

The algorithm will try to simplify the set $\{\mathsf{v}_1, \mathsf{v}_2, \mathsf{v}_5\}$ *in* S *with the set* $\{\mathsf{v}_1, \mathsf{v}_2, \mathsf{v}_3\}$ *computed in Line 7. Note that* CON *with input* $\{\mathsf{v}_1, \mathsf{v}_2\}$ *outputs* v_3. *Therefore, since all the checks will be satisfied in Line 8,* HORN* *will replace* $\{\mathsf{v}_1, \mathsf{v}_2, \mathsf{v}_5\}$ *in* S *with* $\{\mathsf{v}_1, \mathsf{v}_2\}$. *Later, the algorithm will generate the hypothesis* $\mathcal{H} = \{(\mathsf{v}_1 \wedge \mathsf{v}_2 \rightarrow \mathsf{v}_3)\}$ *and at the next equivalence query,* HORN* *will terminate with output* $\mathcal{H}$ *that is logically equivalent to the target.* $\triangleleft$

Simulating Oracles

We recall that the learning framework into consideration is $\mathfrak{F} \coloneqq (\mathcal{E}, \mathcal{L}, \mu)$, where $\mathcal{E}$ is a set of partial interpretations for the Horn language $\mathcal{L}$ and μ is the satisfiability relation between (partial) interpretations and Horn formulas (learning from partial interpretation setting). We propose to extract rules encoded in black-box models with the HORN* algorithm where the trained models act as oracles. The chosen black-box model is a neural network denoted N.

Given the set of variables V, we assume that N is a function $N : \{\top, \bot, ?\}^{|\mathsf{V}|} \rightarrow \{0, 1\}$ that encodes a target Horn theory $\mathcal{T}_N \in \mathcal{L}$. Therefore, N takes as input a partial interpretation $\mathcal{I}$ and it outputs 0 if the encoded target Horn theory $\mathcal{T}_N$ is falsified by $\mathcal{I}$ and 1 otherwise, that is if $\mathcal{I} \models \mathcal{T}_N$. We also assume an arbitrary but fixed ordering on V such that we can express $\mathcal{I}$ as a $\mathsf{vector}(\mathcal{I})$, that is a vector of assignment values. Example 5.9 clarifies the notation.

Example 5.9. *Assume we have the set of variables* $\mathsf{V} = \{\mathsf{v}_1, \cdots, \mathsf{v}_5\}$. *The interpretation*

$\mathcal{I} = \{(\mathsf{v}_1, \top), (\mathsf{v}_2, ?), (\mathsf{v}_3, \bot), (\mathsf{v}_4, \bot), (\mathsf{v}_5, \top}$ *can be represented in vector form as:*

$$\mathsf{vector}(\mathcal{I}) = (1, 0, 0, \;\; 0, 0, 1, \;\; 0, 1, 0, \;\; 0, 1, 0, \;\; 1, 0, 0).$$

Each variable truth value is represented as a triple with the first position set to 1 *if* $\mathcal{I}$ *maps it to* $\top$*, second position set to* 1 *if* $\mathcal{I}$ *maps it to* $\bot$*, and third position set to* 1*if* $\mathcal{I}$ *maps it to* ?*, otherwise* 0*. We assigned the ordering of variables in* V *that follows the ordering of the subscripts of variables in* V*.* ◁

To simulate the membership oracle $\mathsf{MQ}_{\mathfrak{F},\mathcal{T}_N}$, we directly use the classifier N. Whenever Algorithm 7 calls $\mathsf{MQ}_{\mathfrak{F},\mathcal{T}_N}$ with input a partial interpretation $\mathcal{I}$, we check if $N(\mathsf{vector}(\mathcal{I})) = 1$, which means that $\mathcal{I} \models \mathcal{T}_N$. If so, we return the answer 'yes' to the algorithm, 'no' if $\mathcal{I} \not\models \mathcal{T}_N$.

Simulating an equivalence query oracle $\mathsf{EQ}_{\mathfrak{F},\mathcal{T}_N}$ is not as straightforward as simulating $\mathsf{MQ}_{\mathfrak{F},\mathcal{T}_N}$. This is due to the need of checking if the hypothesis constructed is equivalent to $\mathcal{T}_N$. Such query requires evaluating whether the hypothesis constructed by HORN^* is equivalent to $\mathcal{T}_N$. Unfortunately, we already know that there is no algorithm that can always find a counterexample in polynomial time, if one exists. That would imply Horn formulas are polynomially learnable with only membership queries, which is known to not be the case [Angluin, 1988].

For this reason, we drop the requirement of exactly identifying the target Horn theory in favour of finding an approximation of it in light of the PAC learning model (PAC Section 2.4 in Chapter 2). We simulate $\mathsf{EQ}_{\mathfrak{F},\mathcal{T}_N}$ by generating a set of examples randomly and classifying the examples using membership queries (by calling the ANN N). Then, we search for an example in this set that the hypothesis constructed by HORN^* misclassifies. With this strategy, if the size of the set of examples randomly generated is large enough [Vapnik and Chervonenkis, 1971], one can ensure that when the learning process terminates, the generated hypothesis respects PAC conditions. That is, with high probability the total number interpretations misclassified by the hypothesis is below a predefine threshold (considering the entire space of partial interpretations). More precisely, if $\mathcal{L}$ is the language used to define the target $\mathcal{T}_N$, and if the size of the set of examples generated randomly is at least

$$\frac{1}{\epsilon} log_2 \left(\frac{|\mathcal{L}|}{\delta} \right),$$

then one can ensure that the hypothesis constructed is probably approximately correct [Valiant, 1984]. The parameter $\epsilon \in (0, 1)$ indicates the probability that the hypothesis misclassifies an interpretation with respect to the target and $\delta \in (0, 1)$ is the

probability that the learned hypothesis errs more than ϵ. Horn logic is closed under intersection. This means that if $\mathcal{I}$ and $\mathcal{I}'$ satisfy a Horn theory then $\mathcal{I} \cap \mathcal{I}'$ also does [Horn, 1951]. Thus, if $\mathcal{L}$ is restricted to Horn formulas only expressible with Horn logic and variables V, then the number of logically different hypotheses in $\mathcal{L}$ is approximately

$$2^{\binom{|\mathsf{V}|}{\lfloor |\mathsf{V}|/2 \rfloor}}.$$

This number follows from the work done by Alekseev [1989] and Burosch et al. [1993]. This bound is not practical as it would be infeasible to generate a sample respecting PAC conditions with a large set of variables V.

Therefore, in order to guarantee that the hypothesis is PAC, we use the method proposed by Angluin (1988) (Section 2.4) where the size of the sample is not fixed but increases the more equivalence queries are asked. As shown by Equation (2.2) in Section 2.4 (Chapter 2) the size of the sample from which counterexamples are searched in after the i-th equivalence query, should be greater or equal to

$$\lceil \frac{1}{\epsilon} \left(ln \frac{1}{\delta} + i \; ln \; 2 \right) \rceil. \tag{5.1}$$

We will refer to this number when talking about the size of the sample for simulating equivalence queries in the next sections.

Non-Horn Oracles

The truth table associated to the predictions made by a neural network (playing the role of an oracle) may not always reflect the truth table of a Horn theory. This happens when the neural network does not encode Horn rules. Even if the theory used to classify the examples used to train a neural network is Horn, this does not mean that the neural network will have a Horn theory encoded in it. Indeed, as oracles are assumed to answer truthfully, HORN^* would misbehave when answers to queries are returned according to a non-Horn oracle. Example 5.10 shows a case where the HORN^* algorithm may not terminate when the target formula is not Horn.

Example 5.10. Assume $\mathsf{V} = \{\mathsf{v}_1, \mathsf{v}_2\}$ and let N be a trained model that encodes the formula $\mathcal{T}_N \coloneqq \{(\mathsf{v}_1 \vee \mathsf{v}_2)\}$, which is not a Horn KB. Therefore, N classifies the interpretation $\mathcal{I}_1 \coloneqq \{(\mathsf{v}_1, \bot), (\mathsf{v}_2, \bot)\}$ as 0 and the interpretations $\mathcal{I}_2 \coloneqq \{(\mathsf{v}_1, \top), (\mathsf{v}_2, \bot)\}$, $\mathcal{I}_3 \coloneqq \{(\mathsf{v}_1, \bot), (\mathsf{v}_2, \top)\}$, and $\mathcal{I}_4 \coloneqq \{(\mathsf{v}_1, \bot), (\mathsf{v}_2, \top)\}$ as 1. This means that N encodes a theory *not* closed under intersection (and, thus, not Horn).

Following the steps of Algorithm 7, the HORN^* algorithm starts with the empty hypoth-

esis and it asks an equivalence query. $\mathcal{I}_1$ may be returned as a negative counterexample in Line 6 and the algorithm will then add the empty set to S in Line 11 (S is empty at the first iteration). Then, as $\mathcal{T} \not\models \mathsf{v}$ for any $\mathsf{v} \in \mathsf{V}$, the algorithm will never add any rule to $\mathcal{H}$ in Line 6, and it will end the main loop with again the empty hypothesis. The algorithm may receive in Line 6 again the counterexample $\mathcal{I}_1$ after a new equivalence query. Therefore, the algorithm can loop indefinitely without improving the hypothesis when the target is not a Horn theory. ◁

A first solution would be to avoid returning counterexamples more than once, but this may lead to some problems because of how the HORN^* learning algorithm works. Example 5.11 clarifies the need of returning seen counterexamples in previous equivalence queries for the correct execution of Algorithm 7.

Example 5.11. Assume $|\mathsf{V}| = 4$ and the target is

$$\mathcal{T} = \{((\mathsf{v}_1 \wedge \mathsf{v}_3) \rightarrow \mathsf{v}_4), (\mathsf{v}_1 \rightarrow \mathsf{v}_2)\}.$$

Suppose the algorithm asks the first equivalence query and it receives the negative counterexample $\mathcal{I}_1 \coloneqq \{(\mathsf{v}_1, \top), (\mathsf{v}_2, \top), (\mathsf{v}_3, \top), (\mathsf{v}_4, \bot)\}$, and after a second equivalence query, another negative counterexample $\mathcal{I}_2 \coloneqq \{(\mathsf{v}_1, \top), (\mathsf{v}_2, \bot), (\mathsf{v}_3, \bot), (\mathsf{v}_4, \top)\}$.

At this point, the list of antecedents kept by the algorithm would be $S = (\{\mathsf{v}_1\})$. After some positive counterexamples, the hypothesis generated by the algorithm could then be $\{\mathsf{v}_1 \rightarrow \mathsf{v}_2\}$, and the only negative counterexample that can be received to learn the other clause in the target would be again $\mathcal{I}_1$. Therefore, it is important for the correct execution of the algorithm to receive already seen counterexamples. ◁

In order to avoid the loop in Example 5.10, the learning algorithm checks, after a counterexample has been processed, if every set of variables in the list S contributes to the addition of at least one rule in the hypothesis. By construction, if this check is not passed we find ourselves in the case of Example 5.10, we can ignore that counterexample in future iterations. This means that we do not return such counterexample to the algorithm if we find it in the sample of interpretations in future equivalence queries.

Representing Constraints

We explain how we can express constraints that are going to be extracted in the experimental section. Horn rules r are of the form $\mathsf{ant}(r) \rightarrow \mathsf{con}(r)$, for example:

$$(\mathsf{sunny} \wedge \mathsf{happy}) \rightarrow \mathsf{jogging}$$

where all the variables both in the antecedent and in the consequent are not negated. This means that with Horn logic we cannot express rules of the form:

$$(\neg \mathsf{sunny} \wedge \mathsf{happy}) \rightarrow \mathsf{boardgame\_night}.$$
$$(\mathsf{empty\_fridge} \wedge \mathsf{hungry}) \rightarrow \neg \mathsf{happy}.$$

To express a 'weak' form of negation, we duplicate all the variables in V and treat every new variable as the negation of a variable in V. For example, let $\overline{\mathsf{v}}_i$ be the duplicated variable of any $\mathsf{v}_i \in \mathsf{V}$. We can express the rules

$$(\overline{\mathsf{sunny}} \wedge \mathsf{happy}) \rightarrow \mathsf{boardgame\_night}.$$
$$(\mathsf{empty\_fridge} \wedge \mathsf{hungry}) \rightarrow \overline{\mathsf{happy}}.$$

Usually, when duplicating variables in this way, we would like to avoid that both paired variables are true in a partial interpretation (since they represent each other's negation). For this reason, we assume that Horn rules of the form

$$(\mathsf{v} \wedge \overline{\mathsf{v}}) \rightarrow \bot \tag{5.2}$$

always hold, for every $\mathsf{v} \in \mathsf{V}$. The constraint in Equation (5.2) for each variable v is added to the hypothesis before the learning algorithm HORN^* starts.

5.2 Experiments

In this section we show experimental results using the approach presented in the previous section where a trained neural network is treated as an oracle for the HORN^* algorithm. We implemented the algorithm in a Python 3.9 script. For the neural networks, we used the Keras library [Chollet et al., 2015]. Our HORN^* implementation can start with an empty hypothesis or with a set of Horn formulas as background knowledge (assumed to be true properties of the domain at hand). We conduct the experiments on an Ubuntu 18.04.5 LTS server with i9-7900X CPU at 3.30GHz, 32 physical cores, 8 GPUs NVIDIA A100 with 80GB, and 32GB RAM. For each proposed dataset, we first train the neural network N on partial interpretations (more specifically, their vector representation) classified as 0, if it is a negative example, or 1, if it is a positive example. Then, we run the HORN^* algorithm, which poses queries to N in order to extract the rules encoded in N.

HCC Dataset

We first experiment our approach of extracting Horn theories from partial interpretations on a dataset (HCC) in the medical domain [Santos et al., 2015]. This dataset contains missing values for attributes. We can consider each instance as a partial interpretation that sets some variables (attributes of that instance) to true, some to false, and other variables to 'unknown'. Hepatocellular carcinoma causes liver cancer, and it is a serious concern for global health. The HCC dataset [Santos et al., 2015] consists of 165 instances of many risk factors and features of real patients diagnosed with this illness.

There are 49 features selected according to the EASL-EORTC (European Association for the Study of the Liver - European Organisation for Research and Treatment of Cancer). From these features, 26 are quantitative variables, and 23 are qualitative variables. Missing values represent 10.22% of the whole dataset and only 8 patients have complete information in all fields (4.85%). The target class of each patient is binary. Each patient is classified positively if they survive after 1 year of having been diagnosed with HCC, and negatively otherwise. 63 cases are labelled negatively (the patient dies) and 102 positively (the patient survives). Quantitative variables describe, for example, the amount of oxygen saturation in the human body, the concentration of iron in the blood, or number of cigarettes packages consumed per year. The range of the values that each variable can assume varies, but it is specified. Qualitative variables can only have two different values in this dataset (either 0 or 1). Usually they describe categorical information such as if the patient comes from an endemic country, or if it is obese, etc.

The HORN* algorithm expects to receive counterexamples in the form of a partial interpretation that specifies the truth values of boolean variables. For this reason, we encode quantitative variables in a binary representation format. The interval of values of each quantitative variable is partitioned into three sub-intervals. These intervals divide the values of the quantitative variable into 'low', 'middle', and 'high' values. For example, the interval of values of the variable that describes the number of cigarettes packages consumed by the patient per year is $[0, 510]$ can be partitioned into $[0, 50], (50, 200], (200, 510]$. The binarisation technique follows the same idea explained in Example 5.1.

The binarised dataset has in total $26 * 3 + 23 + 1 = 102$ variables and it can be considered a set of partial interpretations. A missing value in the new dataset is denoted with '?' similarly as in the original one, otherwise the value is 1 (0) if the variable is set to true (false). Each partial interpretation $\mathcal{I}$ matches a rule (not necessarily Horn) of the form

$$(l_1 \wedge \cdots \wedge l_{n-1}) \rightarrow l_n \tag{5.3}$$

haemoglobin_low	hemoglobin_med	hemoglobin_high	smoker	adult	survives
1	0	0	1	?	0

Table 5.3: A simplified description of a patient.

where each l_i is a positive literal if the variable i is set to true in $\mathcal{I}$ and negative otherwise. The literal l_k is not present in the rule if l_k has a missing value (Example 5.12).

Example 5.12. *Let the row in Table 5.3 describe the information of a patient who does smoke, that has a low amount of haemoglobin, and that does not survive after* 1 *year of being diagnosed with HCC. We can express this information under the form of the following rule:* $((\mathsf{haemoglobin\_low} \wedge \neg\mathsf{haemoglobin\_med} \wedge \neg\mathsf{haemoglobin\_high} \wedge \mathsf{smoker}) \rightarrow \neg\mathsf{survives})$. *Variables in the antecedent that do not provide meaningful information can be removed. For instance, the literals* $\neg\mathsf{haemoglobin\_med}$ *and* $\neg\mathsf{haemoglobin\_high}$ *in the presence of* haemoglobin_low *can be removed during data preprocessing. The information stating whether the patient is an adult is missing.* ◁

As explained in the previous section, by duplicating the number of variables and pairing them such that one represents the negation of another variable, we can express the previous rule with a Horn formula. For this reason, we further modify the dataset by duplicating variables. Each new variable semantically represents the negated concept of its paired variable. So, we form a dataset D of partial interpretations with 204 variables.

We can express each example in D with Horn rules like in Formula 5.3. We denote by $\mathcal{T}$ the set of such rules that can be formed by looking at all partial interpretations in the extended dataset. To express disjointness constraints between paired variables, we add to $\mathcal{T}$ also the additional Horn rules of the form $(\mathsf{v}_i \wedge \overline{\mathsf{v}_i}) \rightarrow \bot$ (Formula 5.2). Finally, the dataset used for training the neural network is formed by randomly generating partial interpretations (with 204 variables) whose classification label is 0 if they do not satisfy a rule in $\mathcal{T}$, and 1 otherwise.

Model selection

By only randomly generating partial interpretations (with 204 variables), we can create an unbalanced dataset with most partial interpretations classified as positive by the target Horn theory $\mathcal{T}$ (note: $\mathcal{T}$ is defined above). We solve this problem by oversampling interpretations with negative label that are created by violating rules that match interpretations in the binarised dataset. In total, there are 200 negative examples and 200 positive examples in the training dataset. 80% of the (balanced) binarised dataset was used for training and validation. We used 3-fold cross validation for model evaluation. As $\mathcal{T}$ is a Horn theory, there is no noisy data generated in this process.

Hidd. Layers	L. rate	Accuracy
64, 32, 32,16	0.01	0.9252
64, 32, 32, 8	0.01	0.9241
32, 32, 32, 16	0.001	0.9138
32, 32, 16, 8	0.01	0.9159

Table 5.4: Architecture and learning rate of the top four neural networks in ascending order with respect accuracy. The model in the first row was the selected one.

We built a sequential neural network model, where the number of nodes in the input layer is 204, which is the number of variables in an input partial interpretation. We used the library "Keras version 2.4.3" [Chollet et al., 2015] and we empirically searched for the sequential architecture with the best performance varying the number of hidden layers, nodes in hidden layers and the learning rate. We use SGD (Algorithm 3) to optimise the parameters.

We searched our model with the following hyper-parameters:

- number of hidden layers ranging in $\{2, 3, 4, 5\}$;
- number of nodes per layer in $\{4, 8, 16, 32, 64, 128\}$
- learning rate in $\{0.001, 0.01, 0.1\}$.

The model with the best performance had 5 hidden layers, $64, 32, 32, 16$ nodes per layer respectively, and 0.01 learning rate. Table 5.4 shows the best performing architectures.

Test Setting

In our experiments, we run the HORN^* algorithm and we set a limit of 100 equivalence queries that the algorithm can ask before terminating with the built hypothesis as its output. To simulate an equivalence query, we randomly generate a sample of partial interpretations and we classify each interpretation using the neural network. Afterwards, we search for a counterexample to return to the algorithm as the answer of the query. The size of the set of random interpretations for simulating the i-th equivalence query is the same as in Formula 5.1 (Formula 2.2 in Section 2.4 of Chapter 2):

$$s_i := \lceil \frac{1}{\epsilon} \left(ln \frac{1}{\delta} + i \; ln \; 2 \right) \rceil$$

with ϵ and δ set to 0.05. We compare the quality of the HORN^* hypothesis with the hypothesis formed by an incremental decision tree [Domingos and Hulten, 2000b], an established white box machine learning model. We use 'Hoefffding Decision Tree' implementation present in the 'skmultiflow' framework [Montiel et al., 2018]. It is possible to

#Equiv.	$\mathcal{T}/\mathcal{H}$	$\mathcal{T}/N$	$\mathcal{H}/N$	$\mathcal{T}$/tree
100	5.4%	3.1%	4.7%	4.6%

Table 5.5: The outcome of the rule extraction process with the HCC dataset. The numbers are the percentages of interpretations classified differently between the target $\mathcal{T}$, neural network N, HORN* hypothesis $\mathcal{H}$, and the tree.

generate a set of propositional rules by visiting every branch of the tree from the root to leafs labelled negatively. The sampling idea for finding negative counterexamples for HORN* is also used for extracting a decision tree from the neural network. For training the tree, we generate partial interpretations randomly that are then classified by the neural network. The size of the i-th sample generated in this way is s_i. We check if at least one of those classified partial interpretations is misclassified by the decision tree algorithm. If this is the case, we incrementally train the tree with the entire sample. This process is repeated until all classified interpretations in the sample are correctly classified by the tree.

When the HORN* hypothesis and the tree have been extracted, we compute a partial truth table of 204 variables of size $2000s_{100}$. We classify these interpretations according to the target $\mathcal{T}$, the neural network N, the HORN* hypothesis $\mathcal{H}$ and the decision tree. We then compare the truth tables and count the number of times an interpretation is classified differently between the different models.

Results

Table 5.5 shows the outcome of our experiment. The columns $\mathcal{T}/\mathcal{H}$, $\mathcal{T}/N$, $\mathcal{H}/N$, $\mathcal{T}$*/tree* are, respectively, the percentage of interpretations that are labelled differently between the target and the hypothesis, the target and the neural network, the hypothesis and the neural network, and the tree and the target. The running time of the HORN* algorithm with at most 100 equivalence queries was around 6 minutes. The time for extracting an incremental decision tree is around 30 minutes.

The type of rules that the HORN* algorithm extracted are of the form:

$$(\mathsf{hemoglobin\_high} \wedge \mathsf{leucocytes\_low} \rightarrow \mathsf{survives})$$

$$(\mathsf{hemoglobin\_low} \wedge \mathsf{AFP\_high} \wedge \mathsf{ferritin\_medium} \rightarrow \overline{\mathsf{survives}})$$

$$(\mathsf{hemoglobin\_low} \wedge \mathsf{AFP\_high} \wedge \mathsf{ferritin\_high} \rightarrow \overline{\mathsf{survives}})$$

with around 40 different variables in the antecedent. With 100 equivalence queries, the hypothesis extracted has 20 rules of this type that are also present in the target $\mathcal{T}$. Other rules that are entailed by $\mathcal{T}$ can be found in the hypothesis. Examples labelled

negatively with many missing values contain more information about the dependency between variables that must be respected. Indeed, we noticed an increase of the accuracy of the neural network trained on more missing values ensuring balanced classes. As a consequence, also the quality of the extracted rules improves.

Randomly Generated Formulas

We tested this approach also with synthetic datasets. We generate Horn theories $\mathcal{T}$ randomly, with at most n variables and having between $\lfloor \frac{n}{2} \rfloor$ and n clauses. Each variable is treated as an attribute with two values (true, false). We randomly generate interpretations that are labelled 1 if they satisfy $\mathcal{T}$, and 0 otherwise. The dataset contains all possible interpretations with the given variables. As described earlier, the size of the sample for simulating the i-th equivalence query is

$$s_i \coloneqq \lceil \frac{1}{\epsilon}(ln\frac{1}{\delta} + i\ ln\ 2) \rceil.$$

with ϵ and δ set to 0.05 (Equation (2.2) in Section 2.4 of Chapter 2).

Test Setting

In our experiments with the synthetic dataset, we generate a target Horn formula randomly, train the neural network, and run the HORN^* algorithm. 80% is used for training and we use the same model as in the previous subsection. The model complexity of the ANN is adapted to the number of input variables n. Each hidden layer contains half the nodes of the previous layer (rounded up) with a minimum number of nodes of 16. So in total, the ANN has $\lceil log_2(n) \rceil - 4$ hidden layers and 1 node in the output layer. For instance, if the number of variables is 1 000, the ANN has an input layer with 1 000 nodes, and the hidden layers have $500, 250, 125, 63, 32, 16$ nodes, respectively. With this choice, we ensured the accuracy of the neural network to be at least 95% on the validation data. The input is binarized with standard one-hot encoding techniques.

We compare our approach with incremental decision trees [Domingos and Hulten, 2000b], an established white box machine learning model. We use the implementation available in the 'skmultiflow' framework [Montiel et al., 2018]. By visiting the entire built tree, it is possible to generate a set of Horn rules. To train the tree, we employ a similar method we use to extract rules from a neural network using the sampling strategy. We generate a sample of random interpretations classified by the neural network and we test if at least one of the examples is misclassified by the tree. If this is the case, we give the entire sample to the tree as training data and we repeat the process until all classified

examples in the generated sample agree with the tree classification.

We limit to 100 the number of equivalence queries that HORN* is allowed to ask and study how the number of variables affects the outcome of the learning process. The columns $\mathcal{T}/\mathcal{H}$, $\mathcal{T}/N$, $\mathcal{H}/N$ have the same meaning as in the experiments with the HCC dataset: the percentage of interpretations that are labelled differently between the hypothesis, the target, and the neural network. $\mathcal{T}/tree$ is the same percentage between the classification of the built tree and the target. In these tests, we also keep track of the number of predictions *pred* that the neural network provides asked by the HORN* algorithm. We repeat this test also for the case where labels in the training data have been flipped with probability 5% and 10%, that is when there is noise in the data. We test also the scalability of this approach with a number of variables of at most 10 000. We compute the number of interpretations labelled differently from a random sample of size

$$10\, n\, s_{100} = 10\, n\, \lceil \frac{1}{\epsilon}(ln\frac{1}{\delta} + 100\ ln\ 2) \rceil.$$

Results.

Table 5.6 shows $\mathcal{T}/\mathcal{H}$, $\mathcal{T}/N$, $\mathcal{H}/N$, $\mathcal{T}/tree$ and *pred*. These numbers are the average of 10 different iterations for each configuration. In our experiments, $\mathcal{T}/\mathcal{H}$ is at most 7%, that is the algorithm succeeds in finding an approximation the target, even though the neural network misclassifies some interpretations and noise is applied in some cases. With 100 variables, the running time of the HORN* algorithm is on average 2 minutes, while the training time of the decision tree circa 4 minutes. With 10 000 variables, the running time of the HORN* and tree algorithm is respectively of 2 hours, and 4 hours. Table 5.7 shows the time requirements in seconds.

5.3 Dealing With Uncertainty

Uncertainty and incomplete data are common in location-based services, sensor monitoring, social networks, and many other domains. Learning rules based on the information given by imprecise sensors or from data with missing values, in some cases requires a different approach that is able to express uncertainty of rules. Consider for example the task of identify a faulty component of a motor from the inexact measurement of the other parts. Possibility and necessity measures are well suited to model uncertainty in such cases. In Chapter 2, we have shown how to express formulas in possibilistic logic that quantify how certain a piece of knowledge it is. Also, we showed how possibility and necessity measures naturally represent *imprecise* probabilities [Dubois and Prade,

Var.	Noise	$\mathcal{T}/\mathcal{H}$	$\mathcal{T}/N$	$\mathcal{H}/N$	$\mathcal{T}$/tree	Pred.
10	0%	1%	1%	0.5%	3%	1.1e4
	5%	1.1%	1.1%	0.6%	3.1%	1.2e4
	10%	2%	1.2%	1.5%	3.2%	1.2e4
10^2	0%	3.1%	3%	2.9%	3.1%	3.5e4
	5%	3.5%	3.4%	3.1%	3.4%	3.7e4
	10%	3.7%	4.6%	3.3%	3.5%	4.8e4
10^3	0%	5.3%	4.8%	4.2%	4.9%	9.2e4
	5%	5.4%	4.8%	4.3%	5.1%	1.1e5
	10%	5.4%	4.9%	4.3%	5.1%	1.2e5
10^4	0%	6.7%	6.5%	6.2%	6.8%	2.4e6
	5%	6.9%	7%	6.5%	7.1%	4.1e6
	10%	7%	7.2%	6.8%	7.1%	5.4e6

Table 5.6: Rule extraction results with the synthetic dataset and sampling strategy for simulating an equivalence query. The table shows the percentage of interpretations that are labelled differently between the hypothesis, target, neural network, and decision tree.

Var.	HORN*	Tree
10	16	19
10^2	120	230
10^3	1841	2957
10^4	7145	14591

Table 5.7: Running time in seconds of the extraction procedures with synthetic datasets with at most 100 equivalence queries.

1988, 1992], where the possibility value of a formula represents the upper bound of the associated probability value and the necessity value is the lower bound. In this section, we introduce Π_HORN*, an exact learning algorithm for identifying possibilistic Horn formulas. It asks possibility and equivalence queries, and it runs in polynomial time with respect to the size of the target and the number of variables into consideration. Additionally, we discuss how to extract possibilistic Horn rules from black-box machine learning models.

5.3.1 Learning Possibilistic Horn Formulas

We extend some and introduce new definitions presented in Section 2.4 of Chapter 2 that are relevant to our learning setting.

The learning framework $\mathfrak{F} = (\mathcal{E}, \mathcal{L}, \mu)$ in the previous section was defined with $\mathcal{E}$ being a set of partial interpretations over V, $\mathcal{L}$ the Horn logic language, and $\mu(\mathcal{H}) = \{\mathcal{I} \in$

$\mathcal{E} \mid \mathcal{I} \models \mathcal{H}\}$. Until the end of this chapter, by *possibilistic extension of* $\mathfrak{F}$ we mean the learning framework $\mathfrak{F}_\pi = (\mathcal{E}_\pi, \mathcal{L}_\pi, \mu_\pi)$ where $\mathcal{E}_\pi$ is the set of pairs $(\mathcal{I}, \alpha)$ with $\mathcal{I}$ a partial interpretation and $\alpha \in [0,1]$, $\mathcal{L}_\pi$ is the set of possibilistic Horn rules. We say that $(\mathcal{I}, \alpha) \in \mathcal{E}_\pi$ is a positive example for $\mathcal{H} \in \mathcal{H}_\pi$ if $\alpha < \pi_{\mathcal{H}}(\mathcal{I})$. That is, by Point 2 of Lemma 5.14, $(\mathcal{I}, \alpha)$ is a positive example iff $\mathcal{I} \models \mathcal{H}^*_{1-\alpha}$, otherwise it is a negative example. Therefore μ_π is the satisfiability relation between the example $(\mathcal{I}, \alpha)$ and the classical knowledge base $\mathcal{H}^*_{1-\alpha}$.

As the definition of the trained model is different from the one introduced earlier, we also need to take into account what types of queries it can answer. The model N treated in this part solves a regression problem as the output is a number in the interval $[0,1]$. Therefore, a call to N cannot be used anymore to simulate a membership query as it does not provide a answer with binary truth value. The answer of N will be a real number in $[0,1]$ representing a possibility value, instead of 1 ('yes') or 0 ('no'). For this reason, we adapt the membership oracle in Section 2.4 to the possibilistic case, named *possibility oracle*. Let $\mathsf{PQ}_{\mathfrak{F}_\pi,\mathcal{H}}$ be the oracle that takes as input some $(\mathcal{I}, \beta) \in \mathcal{E}_\pi$ and outputs $\pi_{\mathcal{H}}(\mathcal{I})$.[2] A *possibility query* is a call to the oracle $\mathsf{PQ}_{\mathfrak{F}_\pi,\mathcal{H}}$.

We first investigate whether polynomial time learnability results in classical logic are transferable to this new setting, where the learner can ask equivalence queries, and membership queries are replaced by possibility queries. If so then, since Horn formulas are exactly learnable in polynomial time, by Corollary 5.7 it would follow that this holds for possibilistic Horn. We recall the result in Chapter 4 stating that in general, transferability from the classical to the possibilistic setting does not hold. The argument trivially follows in the learning form partial interpretation setting.

Theorem 4.21. *There is an FO learning framework* $\mathfrak{F}$ *such that* $\mathfrak{F}$ *is in* ELP(MQ,EQ)*, but* $\mathfrak{F}_\pi = (\mathcal{E}_\pi, \mathcal{L}_\pi, \mu_\pi)$ *is not in* ELP(MQ,EQ)*.*

Fortunately, for the class of safe learning frameworks (Section 4.1 in Chapter 4) which contains the Horn learning framework, transferability from the classical to the possibilistic setting holds as stated by Theorem 5.13.

Theorem 5.13. *Let* $\mathfrak{F}$ *be a safe learning framework.* $\mathfrak{F}_\pi$ *is exactly learnable in polynomial time with possibility and equivalence queries iff* $\mathfrak{F}$ *is exactly learnable in polynomial time with membership and equivalence queries.*

In Theorem 5.13, we can remove the *safe* condition for the direction from $\mathfrak{F}_\pi$ to $\mathfrak{F}$. Based on our result in Theorem 5.13, we adapted the classical HORN* algorithm to the possibilistic setting, To learn possibilistic Horn formulas, we train the neural network with

[2]The precision of this number is finite if $\mathcal{H}$ is finite.

interpretations (more precisely, their vector representations) annotated with a possibility value. A comprehensive proof of Theorem 5.13 is found in Appendix A.3.

5.3.2 The Π_HORN* Algorithm

The positive transferability result shown in the previous section, encouraged the development of a possibilistic version of the HORN* algorithm in the learning from partial interpretation setting. Algorithm 9 depicts the steps that should be taken to learn a hypothesis equivalent to an unknown possibilistic Horn formula $\mathcal{T}$ by asking possibility and equivalence queries, by respectively calling $\mathsf{PQ}_{\mathfrak{F}_\pi,\mathcal{T}}$, and $\mathsf{EQ}_{\mathfrak{F}_\pi,\mathcal{T}}$. In our argument to show correctness of the Π_HORN* algorithm, we use the following properties.

Lemma 5.14. *Let $\mathcal{T}$ be a possibilistic formula and $\mathcal{I}$ an interpretation such that $\mathcal{I} \models \mathcal{T}^*$. The following properties hold:*

1. *$\pi_\mathcal{T}(\mathcal{I}) = \alpha$ implies $1 - \alpha \in \mathcal{T}^v \cup \{0\}$.*
2. *For $\alpha < 1$, $\pi_\mathcal{T}(\mathcal{I}) > \alpha$ iff $\mathcal{I} \models \mathcal{T}^*_{1-\alpha}$.*

Proof.

1. Let $\pi_\mathcal{T}(\mathcal{I}) = \alpha$. If α is 1, the statement holds. If $\alpha < 1$, by definition of $\pi_\mathcal{T}$, there is a formula $(\phi, \beta) \in \mathcal{T}$ where β the highest valuation such that $\mathcal{I} \not\models \phi$. Consequently, $\alpha = 1 - \beta$ and it follows that $1 - \alpha = \beta \in \mathcal{T}^v$.
2. ($\Rightarrow$) Assume $\mathcal{I} \not\models \mathcal{T}^*_{1-\alpha}$. Then, there is $\phi \in \mathcal{T}^*_{1-\alpha}$ such that $\mathcal{I} \not\models \phi$. By definition of $\pi_\mathcal{T}$, we have that $\pi_\mathcal{T}(\mathcal{I}) \leq 1 - (1 - \alpha) = \alpha$.

   ($\Leftarrow$) Assume $\mathcal{I} \models \mathcal{T}^*_{1-\alpha}$. Then, $\mathcal{I}$ satisfies every formula $\phi \in \mathcal{T}^*_{1-\alpha}$. If $\mathcal{I}$ does not satisfy an arbitrary $(\phi, \beta) \in \mathcal{T}$, then β must be smaller than $1 - \alpha$ and by definition of $\pi_\mathcal{T}$, we know $\pi_\mathcal{T}(\mathcal{I}) \leq 1 - \beta$ which is greater than $1 - (1 - \alpha) = \alpha$ because $\beta < 1 - \alpha$. □

Similarly to the HORN* algorithm, Π_HORN* starts with the empty hypothesis and an empty list S that stores antecedents of rules in $\mathcal{T}$ in addition to the possibility value originally assigned to the counterexample revealing such antecedent. Π_HORN* starts its main loop by calling the equivalence query oracle, and it will terminate only when a call to the equivalence query signals that the built hypothesis is equivalent to the target.

Upon receiving a counterexample $(\mathcal{I}, \alpha')$, we know by Lemma 5.15 that $(\mathcal{I}, \alpha') \not\models \mathcal{T}$.

Lemma 5.15. *Let $\mathcal{T}$ be the target possibilistic Horn formula and $\mathcal{H}$ be the hypothesis built by* Π_HORN*. *At all times during the run, $\mathcal{T} \models \mathcal{H}$.*

Proof. At the start of the computation when $\mathcal{H} = \emptyset$, it trivially holds. Moreover, Line 17 is the only place where the algorithm adds rules to the hypothesis. For any rule (ϕ, β) added to the hypothesis, we have that the interpretation $\mathcal{I}^\phi$ such that it maps to $\top$ only variables in the antecedent of ϕ, it maps to $\bot$ the consequent of ϕ, and to '? the remaining variables, then $\mathcal{I}^\phi \not\models \mathcal{T}_\beta$. By Property 2 of Lemma 5.14, only possibilistic rules entailed by $\mathcal{T}$ are added to $\mathcal{H}$ after calls to Algorithm 10. □

Therefore, a subset of variables mapped to $\top$ by $\mathcal{I}$ satisfy the antecedent of a rule in $\mathcal{T}$ that $\mathcal{I}$ falsifies. In addition to keeping track of rule antecedents in the target, Π_HORN* should also compute the possibility α associated to such counterexample with respect to $\mathcal{T}$. This is done by calling the possibility query oracle in Line 7.

Algorithm 9: Π_HORN*

1: **Input:** It is assumed that the learner knows $\mathfrak{F}_\pi$ but not the target $\mathcal{T}$.
2: **Output:** $\mathcal{H}$ such that $\mathcal{H} \equiv \mathcal{T}$.
3: Let S be the empty sequence.
4: Denote with $(\mathcal{A}_i, \alpha_i)$ the i-th element of S.
5: Let $\mathcal{H}$ be the empty hypothesis.
6: **while** $\mathsf{EQ}_{\mathfrak{F},\mathcal{T}}(\mathcal{H})$ returns a counterexample $(\mathcal{I}, \alpha')$ **do**
7: $\quad\alpha \leftarrow \mathsf{PQ}_{\mathfrak{F}_\pi,\mathcal{T}}(\mathcal{I})$
8: $\quad\hat{\mathcal{A}} \leftarrow \{\mathsf{v} \in \mathsf{V} \mid (\mathcal{H} \wedge \{\mathsf{v}' \in \mathsf{V} \mid \mathcal{I}(\mathsf{v}') = \top\}) \models (\mathsf{v}, 1-\alpha)\}$
9: $\quad$**if** there is $\mathcal{A}_i \in S$ such that $\mathcal{A}_i \cap \hat{\mathcal{A}} \subset \mathcal{A}_i$ and Π_CON$(\mathcal{A}_i \cap \hat{\mathcal{A}}, 1-\alpha) \neq \emptyset$ **then**
10: $\quad\quad$replace the first such $(\mathcal{A}_i, \alpha)$ with $(\mathcal{A}_i \cap \hat{\mathcal{A}}, \alpha_i)$ in S
11: $\quad$**else**
12: $\quad\quad$append $\hat{\mathcal{A}}$ to S
13: $\quad$**end if**
14: $\quad\mathcal{H} \leftarrow \emptyset$
15: $\quad$**for** $(\mathcal{A}, \alpha) \in S$ **do**
16: $\quad\quad$**for** $\mathsf{u} \in \Pi$_CON$(\mathcal{A}, 1-\alpha)$ **do**
17: $\quad\quad\quad$add the rule $((\bigwedge_{\mathsf{v} \in \mathcal{A}} \mathsf{v}) \rightarrow \mathsf{u}, 1-\alpha)$ to $\mathcal{H}$
18: $\quad\quad$**end for**
19: $\quad$**end for**
20: **end while**
21: **return** $\mathcal{H}$

By Points 1,2 of Lemma 5.14, knowing the possibility value, and the information that $(\mathcal{I}, \alpha') \not\models \mathcal{T}$ reveal a valuation in the target. Later in the algorithm, the value α associated to a set of variables will be used to assign the necessity degree to formulas in the hypothesis.

Algorithm 10: Π_CON

1: **Input:** variables $\mathcal{A} \subseteq \mathsf{V}$, possibility α
2: **Output:** A subset $\mathcal{A}'$ of $\mathsf{V} \cup \{\bot\}$ such that $\mathcal{T} \models (\mathcal{A} \to \mathsf{v}, \alpha)$ for all $\mathsf{v} \in \mathcal{A}'$
3: $\mathcal{A}' \leftarrow \emptyset$
4: **for** $\{\mathsf{u} \mid \mathsf{u} \in \mathsf{V} \cup \{\emptyset\} \setminus \mathcal{A}\}$ **do**
5: Let $\mathcal{I}$ be such that for $\mathsf{v} \in \mathsf{V}$, $\mathcal{I}(\mathsf{v}) \leftarrow \begin{cases} \top & \mathsf{v} \in \mathcal{A} \\ \bot & \mathsf{v} = \mathsf{u} \\ ? & \text{otherwise} \end{cases}$
6: **if** $\mathsf{PQ}_{\mathfrak{F}_\pi,\mathcal{T}}(\mathcal{I}) \leq \alpha$ **then**
7: add u to $\mathcal{A}'$
8: **end if**
9: **end for**
10: **return** $\mathcal{A}'$

After having found the variables that must be mapped to $\top$ according to the knowledge acquired so far in Line 8, Π_HORN* attempts to simplify the antecedent of a rule previously found stored in S. The reason behind each step is analogous to the classical HORN* algorithm, but this time the algorithm should be aware of the different valuations in the target. An element $(\mathcal{A}_i, \alpha_i) \in S$ will be replaced with $(\mathcal{A}_i \cap \mathcal{A}, \alpha_i)$ if there is a rule $(\mathsf{r}, 1-\alpha_i)$ in $\mathcal{T}$ with $\mathsf{ant}(\mathsf{r}) \subseteq \mathcal{A}_i \cap \hat{\mathcal{A}}$. The check in Line 9 guarantees that rules with smaller antecedents are identified first. Intuitively, the algorithm runs the HORN* algorithm for each different valuation in the set of received counterexamples S. Correctness and polynomial time learnability of Algorithm 9 follow by Theorem 5.13.

Theorem 5.16. *Let* V *be a finite set of propositional variables and let* $\mathcal{T}$ *be the unknown target possibilistic Horn formula.* Π_HORN* *runs in polynomial time with respect to the size of the target* $|\mathcal{T}|$*, and number of variables* $|\mathsf{V}|$*, and outputs a hypothesis* $\mathcal{H}$ *such that* $\mathcal{H} \equiv \mathcal{T}$ *by asking possibility and equivalence queries.*

Proof. By Lemma 5.15, Π_HORN* can only receive negative counterexamples. The possibilistic formula $\mathcal{T}$ can be represented as a set of *layered* classical horn formulas $\mathcal{T}^*_{1-\alpha}$ for each $(1-\alpha) \in \mathcal{T}^v$. Therefore, the idea is to learn a each *classical* layer of $\mathcal{T}$ with the HORN* algorithm.

Whenever Π_HORN* receives a counterexample $(\mathcal{I}, \alpha')$ from a call to $\mathsf{EQ}_{\mathfrak{F}_\pi,\mathcal{T}}$, it calls $\mathsf{PQ}_{\mathfrak{F}_\pi,\mathcal{T}}$ to obtain $\alpha = \pi_{\mathcal{T}}(\mathcal{I})$. By Property 1 of Lemma 5.14, we have $\alpha \in \mathcal{T}^v$. Therefore, at most $|\mathcal{T}^v|$ different valuations are returned after a call to the possibility oracle. For each of such valuation α, by Property 2 of Lemma 5.14, Π_HORN* follows the same steps of HORN* to build a hypothesis $\mathcal{H}^{1-\alpha}$ such that $\mathcal{H}^{1-\alpha} \equiv \mathcal{T}^*_{1-\alpha}$.

Lemma 4.9. *Let* $\mathcal{T}$ *be a possibilistic KB. Let* I *be a finite set of valuations such that* $\mathcal{T}^v \subseteq I$*. If for each* $\alpha \in I$ *there is some FO KB* $\mathcal{K}^*_\alpha$ *such that* $\mathcal{K}^*_\alpha \equiv \mathcal{T}^*_\alpha$ *then, it holds*

that $\mathcal{T} \equiv \{(\phi, \alpha) \mid \phi \in \mathcal{K}^*_\alpha, \alpha \in I\}$.

By Theorem 5.6 and Lemma 4.9, we can prove that the statement holds. □

Also in this case, the possibility distribution encoded by the neural network may not reflect a possibilistic Horn theory. We adopt a similar solution for solving the problem of counterexamples not reflecting a Horn theory as in Example 5.10. That is, after a counterexample is received, Π_HORN* checks if every element $(\mathcal{A}, \alpha) \in S$ is responsible for the addition of at least one rule in the hypothesis. If it is not the case, $(\mathcal{A}, \alpha)$ is never returned as a counterexample again.

5.3.3 Experimental Results

We experiment the extraction of possibilistic Horn formulas with a possibilistic synthetic dataset. Similarly as in the classical case, we generate a possibilistic Horn theory $\mathcal{T}$ randomly, with at most n variables and having between $\lfloor \frac{n}{2} \rfloor$ and n clauses. The valuation associated to a rule in $\mathcal{T}$ is chosen randomly in the interval $(0, 1]_2$.[3] We then generate partial interpretations $\mathcal{I}$, also randomly. Each interpretation is classified with the possibility value $\pi_\mathcal{T}(\mathcal{I})$. In this way, we generate a dataset $D \coloneqq \{(\mathcal{I}_i, \pi_\mathcal{T}(\mathcal{I}_i)) \mid 1 \leq i \leq d\}$, where the size depends on the number of variables into consideration, with $d = 3000\, n$.

Similarly as in the previous part of this chapter, we assume to have a trained machine learning model, here a neural network N. This model takes as input a vector of variable assignments $\{\top, \bot, ?\}^{|\mathsf{V}|}$ and it outputs a value in $[0, 1]$, representing the possibility of that assignment with respect to the encoded possibility distribution by N. Assuming that N encodes a possibilistic Horn formula $\mathcal{T}_N$, in this setting, the ANN N with input a partial interpretation $\mathcal{I}$ outputs $\pi_{\mathcal{T}_N}(\mathcal{I})$. The dataset used to train the ANN model is $\{(\mathsf{vector}(\mathcal{I}_i), \alpha_i) \mid 1 \leq i \leq d\}$. Recall that $\mathsf{vector}(\mathcal{I}_i)$ stands for the one-hot transformation of a vector representation of an interpretation (Example 5.9). This is because each attribute is treated as a categorical type with three values (0, 1, ?). In the end, the dataset is a set of one-hot encoded partial interpretations $\mathcal{I}_i$ with label α_i. We train an ANN model such that given as input the encoding $\mathsf{vector}(\mathcal{I})$ of a partial interpretation $\mathcal{I}$, it outputs a number in $[0, 1]$ expressing the possibility associated to $\mathcal{I}$, that is $\pi_{\mathcal{T}_N}(\mathcal{I})$. After training the ANN model, we run the Π_HORN* algorithm. Calls to membership queries with input $(\mathcal{I}, \alpha)$, by Point 2 of Lemma 5.14, will be answered 'yes' if $N(\mathsf{vector}(\mathcal{I})) > \alpha$, 'no' otherwise. So, the ANN directly simulates $\mathsf{PQ}_{\mathfrak{F}_\pi, \mathcal{T}_N}$. Also in this

[3] The subscript in the interval means the precision.

Variables	Hidd. Layers	Learn. rate	MSE
10	64	0.1	0.0194
10^2	128, 64, 64	0.01	0.0291
10^3	512,256,64	0.001	0.0473
10^4	1024,512,512,64	0.001	0.8812

Table 5.8: Architecture and learning rate of the best neural network.

case, to simulate the i-th equivalence query, we create a sample of size

$$s_i = \lceil \frac{1}{\epsilon}(ln\frac{1}{\delta} + i\ ln\ 2) \rceil.$$

with ϵ and δ set to 0.05. We let the trained ANN classify each generated example, and we search for a (negative) counterexample in it. If we found one, we return it to the algorithm, otherwise we stop the run of Π-HORN* and obtain the extracted hypothesis.

Model Selection

80% of the data was reserved for training and validation. We used 3-fold cross validation. We searched for the best combination of hyper-parameters:

- number of hidden layers ranging in $\{5, 10, 15, 20, 25\}$;
- number of nodes per layer in $\{64, 28, 256, 512, 1024\}$
- learning rate in $\{0.001, 0.01, 0.1\}$.

We optimise the parameters with SGD (Algorithm 3) and we set to 2000 the maximum number of epochs with early-stopping. This means that when the validation loss does not decrease more than 0.001 after 10 times the parameter have been updated, the optimisation algorithm stops. Table 5.8 shows the best performing architectures depending on the number of variables into consideration during our tests.

Test Setting

The baseline model of choice is an implementation of an incremental decision tree regressor [Domingos and Hulten, 2000b] available on the 'skmultiflow' framework [Montiel et al., 2018]. To train the tree, we generate a sample of random interpretations classified by the neural network and we test if at least one of the examples is misclassified by the tree. If this is the case, we give the entire sample to the tree as training data. We

Var.	**Noise**	$\mathcal{T}/\mathcal{H}$	$\mathcal{T}/N$	$\mathcal{H}/N$	$\mathcal{T}$/**tree**	**Pred.**
10	0%	0.3%	0.5%	0.3%	0.4%	$1.2e4$
	5%	0.4%	0.5%	0.4%	0.5%	$1.4e4$
	10%	0.5%	0.6%	0.4%	0.5%	$1.5e4$
10^2	0%	0.7%	0.6%	0.7%	0.7%	$3.7e4$
	5%	0.7%	0.8%	0.7%	0.8%	$3.9e4$
	10%	0.8%	0.9%	0.9%	0.8%	$5.1e4$
10^3	0%	1.2%	1.1%	1.3%	1.3%	$9.7e4$
	5%	1.3%	1.3%	1.3%	1.4%	$1.3e5$
	10%	1.5%	1.6%	1.4%	1.6%	$1.3e5$
10^4	0%	2.2%	2.4%	2.3%	2.7%	$2.5e6$
	5%	2.4%	2.4%	2.5%	2.7%	$4.2e6$
	10%	2.8%	2.9%	2.8%	2.9%	$5.5e6$

Table 5.9: Rule extraction results with the synthetic dataset and sampling strategy for simulating an equivalence query. The table shows the percentage of interpretations that are labelled with a difference between possibility values greater than 0.05.

Var.	Π_HORN*	**Tree**
10	348	1021
10^2	420	351
10^3	$6.2e3$	$5.2e4$
10^4	$7.4e4$	$6.1e5$

Table 5.10: Running time in seconds of the extraction procedures with synthetic datasets with at most 100 equivalence queries.

repeat the process until the tree predictions are the same as the ANN. The size of the i-th generated sample is s_i.

We limit Π_HORN* to ask at most 100 equivalence queries. We vary the number of variables from 10 to 10 000. To simulate cases where we have noisy data, we repeat this test also for cases where labels in the training data have been added or subtracted a random value between $[0.1, 0.3]_2$ with probability 5% and 10%; making sure that the corrupted labels are in the interval $[0, 1]$. We record the percentage of interpretations in which the difference between possibility values passed 0.05 when comparing the hypothesis $\mathcal{H}$, the target $\mathcal{T}$, the neural network N, and the tree. We compute the number of interpretations labelled differently from a random sample of size

$$10\, n\, s_{100} = 10\, n\, \lceil \frac{1}{\epsilon}(ln\frac{1}{\delta} + 100\ ln\ 2) \rceil.$$

Results

Table 5.9 shows the outcome of out experiments. The columns $\mathcal{T}/\mathcal{H}$, $\mathcal{T}/N$, $\mathcal{H}/N$, $\mathcal{T}/tree$ is respectively the percentage of interpretations that differ by more than the value 0.05 between the target and the hypothesis, the target and the neural network, the hypothesis the neural network, and the target and the tree. The last column, 'Pred', is the number of predictions that the neural network provides for answering queries to the Π_HORN* algorithm. Table 5.10 shows the time required by Π_HORN* and incremental tree algorithm to extract rules from the neural network. We conclude that the hypothesis generated by Π_HORN* is more accurate with comparable running times.

5.4 Discussion

In this work, we presented an approach for simulating queries, defined in the exact learning model, for cases where it is not realistic to have oracles that can answer queries. The most challenging part is how to simulate equivalence queries, in particular, how to generate counterexamples. We evaluate a strategy for simulating them, based on sampling. It is often the case that not all values in a dataset are known or trustable. Our method based on partial interpretations covers such scenarios and generalizes the case with (full) interpretations. We test our approach empirically using a real world dataset in the medical domain. We tested this method by extracting Horn theories from neural networks, using the HORN* learning algorithm. The quality of the extracted hypothesis and the runtime results are promising. Moreover, there is also some potential for optimising our implementation.

Additionally, data is not always certain or complete. We tackle the problem of learning in the presence of uncertainty, and express knowledge with degrees of confidence using the framework of possibilistic logic and the exact learning model. We prove that polynomial time results can be transferred from classical to possibilistic settings. For this reason, we implement and test Π_HORN* an adaptation of HORN* to the possibilistic case. Π_HORN* is time efficient and effective in identifying the unknown possibilistic target formula. This work contributes to the challenge of how to represent ignorance in black-box machine learning models, like neural networks.

Future works will focus on studying alternative ways of simulating exact learning oracles. Depending on the specific context where an exact learning algorithm is run, there may be new queries that can be performed. For instance, in the possibilistic setting, it was convenient to replace the membership oracle with the possibility one. Moreover,

polynomial time learnability results hold up to multivalued dependency formulas [Hermo and Ozaki, 2020]. Investigating more expressive classical languages, may lead to new methods for learning more complex possibilistic rules.

Chapter 6

Related Work

We give an overview of relevant works related to ours. In Section 6.1, we list pertinent efforts in combining the knowledge representation advantages of logic-based formalisms and performant inductive inference of statistical methods. In Section 6.2, we present important studies in the exact learning of logical theories. Finally, in the last section we discuss proposed formalisms with the goal of modelling uncertainty, and works on learning possibilistic logic theories.

6.1 Rule Induction and Neurosymbolic AI

The most used rule induction algorithms belong to the decision tree family. In particular, the classification and regression tree (CART) algorithm [Breiman et al., 1984] or the ID4.5 [Quinlan, 1993] decision tree algorithm. Such trees are built by splitting the initial dataset, that represents the root node of the tree, into subsets according to the value of a feature. These values then constitute the successor children. This process is repeated on each subset representing each child until new data-subsets are homogeneous, according to some predefined stopping criteria. In literature, we can find many variants based on the idea of a decision tree, such as RIPPER [Cohen, 1995] that generates binary (or crisp) rules. We also recall FURIA, an improvement of RIPPER, that generates fuzzy rules [Elkano et al., 2020; Hühn and Hüllermeier, 2009]. Decision trees are simple to interpret, handle missing data, both categorical and numerical features, and there are efficient implementations of tree learning algorithms that are open source [Pedregosa et al., 2011]. Unfortunately, the complexity of finding an optimal tree is intractable [Hyafil and Rivest, 1976] and techniques based on greedy heuristics make the tree building process dependant on the structure of the data. The issue of scalability in rule induction has al-

ready been observed in Big Data settings. For tackling the problem of learning trees with big datasets, Elkano et al. [2020] proposed CFM-BD, a distributed system for fuzzy rule induction using a MapReduce paradigm. While CFM-BD has shown promising results, it still solves a search problem on a large discrete space. Decision trees mainly differ from the RIDDLE architecture as the type of data given as input (RIDDLE requires possibility values), and decision trees have 'sharp' decision boundaries while with RIDDLE the learning task can be cast into a differentiable error function.

Other approaches aim at finding directly the rules holding in a dataset. Works in inductive logic programming (ILP) like PROGOL [Muggleton, 1995] combine reasoning and search to find the rules to add to a given background knowledge such that the final hypothesis (background + rules) gives correct predictions. However, the applications of such systems is limited by the high computational costs they require to find rules. Another method used to discover patterns in data under the form of rules is association rule mining (ARM) [Agrawal et al., 1993]. With this approach, rules are found looking at correlations, causal structures among sets of features in databases. Generally, the search procedures selects candidate rules and computes the statistical significance, called *support*, and percentage of cases that satisfy a given rule, called *confidence*. When both support and confidence of a rule are above a given threshold, such rule is outputted. The proposed RIDDLE architecture has the main advantage of casting the learning problem into a differentiable error function, so efficient optimization algorithms can be used.

Recently, we witnessed a increase in generating interpretable rules with *Artificial Neural Network* (ANN) models. The goal is to combine the advantages of symbolic logic of carrying reasoning and representing knowledge, with the large-scale learning capabilities of the algorithms used to train ANN models. Hence the reason of the name of this field called neuro-symbolic artificial intelligence. On the most abstract level, works in this field propose different architecture for ANN models that can be trained with standard gradient descent techniques, such that after training, the optimised weights can be used to construct rules. DR-Net [Qiao et al., 2021] employs a simple 2-layer neural network architecture to learn DNF rule sets. DR-Net also controls the complexity of the rules learned via a sparsity term. Kusters et al. [2022] define a 3-layer neural network architecture for rule induction, named R2N, that can also identify potential new terms. R2N integrates neural networks and rule induction with a differentiable function, but it can only learn positive DNF, a restricted class of rules where variables cannot be negated [Angluin, 1988]. Glanois et al. [2022] propose HRI, a hierarchical approach to rule induction. The language of rules differs considerably from propositional rules, and their method also relies of pre-defined rule templates that determine the types of rules that can be learned. The RIDDLE architecture does not limit the number of layers, and it is able to express full propositional CNF rules.

6.2 Exact Learning

The exact learning model [Angluin, 1988] has been applied for investigating and solving problems in computational learning theory. The works by Cohen and Hirsh (1994) and Frazier and Pitt (1996) present results on the learnability of formulas that describe an abstract concept in the exact learning model using membership and equivalence queries. Konev et al. (2014) present results on the exact learning of lightweight description logic ontologies. They make a distinction between polynomial time and polynomial query learnability (where not the running time but only the number and the size of queries asked by the learner are taken into account) [Konev et al., 2016, 2018]. Positive polynomial time learnability results for lightweight description logics have also inspired the implementation of 'ExactLearner' [Duarte et al., 2018], a tool that can build ontologies by asking queries to domain experts. This tool also shows the strength of the algorithm with the implementation of a teacher that returns 'difficult' answers that minimise the information revealed about the target. Among the systems built for learning description logic concepts, we highlight the DL-Learner proposed by Lehmann (2009). Some of the results concerning learnability of lightweight logical theories were presented by Ozaki et al. (2020) where they replaced an equivalence query with an inseparability query that returns 'yes' if the hypothesis is inseparable from the target with respect to a predefined query language.

An important open question in computational learning theory is whether the class of formulas in conjunctive normal form is learnable in polynomial time. Horn logic is efficiently learnable in the exact learning model with membership and equivalence queries [Angluin et al., 1992] and a recent work shows positive results for a more expressive class of propositional logic formulas [Hermo and Ozaki, 2020]. About the exact learning of more expressive logic, we recall works on the exact learnability of fragments of FO Horn logic under certain conditions [Arias and Khardon, 2002; Arimura, 1997a; Reddy and Tadepalli, 1998a; Selman and Fern, 2011]. There is also an implementation that instead of asking queries to domain experts, it just looks at available data to mine a fragment of FO Horn rules, this system is called Logan-H [Arias et al., 2007]. The original algorithm is polynomial in the number of clauses, terms and predicates and the size of the counterexamples, but exponential in the arity of predicates and in the number of variables per clause. The existence of a polynomial time exact learning algorithm for FO Horn with respect to the number of variables per clause is still an open question. A lot of work has been done in the field of learning logical theories. We point out to comprehensive surveys [Cimiano et al., 2010; Lehmann and Völker, 2014; Ozaki, 2020a; Wong et al., 2012]. Regarding works extending Angluin et al.'s (1992) algorithm for learning propositional Horn theories, we highlight the works by Arias and Balcázar [2009]

and Hermo and Ozaki [2020]. Some authors addressed the problem of learning logic theories with only membership queries. Lavín Puente (2011) shows that a fragment of propositional logic where the consequents of the clauses in the target are pairwise disjoint and with a constant maximum number of variables are polynomial time learnable with only membership queries. Monotone CNF formulas [Domingo et al., 1999] in which each variable appears at most a constant number of times are also learnable in polynomial time with only membership queries. Decision trees with a fixed depth are also learnable with only membership queries [Bshouty and Haddad-Zaknoon, 2019].

In literature, we can find other works that use exact learning to extract an interpretable representation of a neural network by posing queries. In 2019, Weiss et al. extend their already mentioned approach for extracting automata from neural networks to the case in which the automata are probabilistic. Recently, Okudono et al. (2020) extended the mentioned work with a regression method for simulating equivalence queries. Shih et al. (2019) consider the problem of verifying binarized neural networks with membership and equivalence queries. The authors extract a binary decision diagram using a SAT solver to answer equivalence queries. The interpretability field is large and there are many approaches to interpret neural networks models Zhang et al. [2021]. There is a global and active approach that focuses on explaining the already trained model as a whole, as opposed to changing the network architecture for interpretability (passive), or explaining through feature studies or correlation (local).

6.3 Uncertainty Measures and Possibilistic Logic

Imprecise and uncertain information is abundant and, in some cases, the only available data. There are many attempts for formally representing uncertainty. Probability theory [Ross, 2014] is the most well known uncertainty formalism. We recall belief function-based evidence theory [Shafer, 2016; Yager and Liu, 2008] that similarly as in possibility theory, there is a function, named belief function, that assigns to every element of the domain into consideration, a value in the interval $[0, 1]$ with the additional constraint that the sum of beliefs on all world views must be 1. An event is more certain when the value associated by the belief function approaches 1. The theory of evidence can also be considered a generalisation of Bayesian theory [Jensen, 2001; Pearl, 1988]. The latter is also an important representation formalism for uncertainty in knowledge. To address the need of both managing uncertain and complex structured knowledge, there are some works that combine logic and uncertainty formalisms. Classical logic is too rigid and fails to model situations where there is contradictory, incomplete information; often encountered and abundant in the real world. We mention probabilistic logic [Guan

and Lesser, 1990; Nilsson, 1986], where the truth of a statement is not binary but associated with a probability value. In this way, reasoning tasks are carried out to find the probability of a statement based on the probability of other given statements. A similar formalism is subjective logic [Jøsang, 2001], well suited for modelling situations where knowledge is mostly dubious.

If instead we want to model the vagueness of statements, Fuzzy logic [Pelletier, 2000; Zadeh, 1965] provides a mathematical framework for encoding vague knowledge and reasoning about it. For example, if we want to model the concept of cold, $-45°$ Celsius may be assigned a higher value of 'coldness' (as a membership to the concept 'cold') if compared with $-1°$ Celsius. When the underlying theory for managing incomplete, or inconsistent knowledge is possibility theory [Dubois and Prade, 1998; Zadeh, 1978] we have possibilistic logic [Dubois and Prade, 1990a, 2014; Dubois et al., 1994; Lang, 2000]. It can model uncertainty numerically or quantitatively and, in general, it gives more freedom to model uncertainty as it for example, allows to neither believe in an event nor in its negation (forbidden by the complement rules in other uncertainty measures like probability). Also, possibility theory is advertised as the simplest and non-trivial formalism for modelling imprecise probability [Coolen et al.; Walley, 1996; Walley and Peter, 1991]. The measures introduced by possibility theory can be seen as upper bounds and lower bounds of ill-known probabilities [Dubois and Prade, 1988, 1990b, 1992]. The belief and plausibility measures defined in evidence theory can also be used to model imprecise probabilities, respectively, the lower and upper bounds.

We can find some works that combine learning and possibilistic logic. We can find works on the PAC learnability of possibilistic logic theories from default rules [Kuzelka et al., 2016]. Moreover, possibilistic logic has been used to reason with default rules [Benferhat et al., 1992]. They propose a method for learning consistent logical theories from noisy default rules. When carrying automated reasoning procedures with default rules, it is important to rank rules such that it is easier to derive plausible conclusions according to the given rank applicable to the situation under concern. Possibilistic logic provides a way of selecting and ordering rules. In inductive logic programming, first-order possibilistic logic has been used to handle exceptions by means of prioritised rules [Serrurier and Prade, 2007]. In statistical relational learning, possibilistic logic has been used as a formal encoding of statistical regularities found in relational data [Kuzelka et al., 2017]. Additionally, they claim that possibilistic models are easier to interpret correctly, as they are stratified classical theories. We recall the work by Kuzelka et al. [2015] stating that possibilistic formulas can encode Markov logic networks. Formal concept analysis has been applied to generate attribute implications with a degree of certainty [Djouadi et al., 2010]. Finally, we point out an extension of version space learning that deals with examples associated with possibility degrees [Prade and Serrurier, 2008].

Chapter 7

Conclusion

Finding patterns in datasets under the form of interpretable rules is beneficial to varied fields [Bratko, 1993; WJ., 1987]. Interpretable models can support decisions in biomedicine [Podgorelec et al., 2002; Scala et al., 2019], and security [Dhanraj et al., 2022; Xu et al., 2018], to name a few. Moreover, uncertainty is pervasive and it is crucial to handle to extract useful knowledge from the available data. The work in this thesis provides a method for finding interpretable patterns in data under the form of rules with the additional goal of managing and expressing uncertainty over the learned knowledge. In this chapter, we summarize the main contribution of this work. Then, we mention open problems and possible future works.

7.1 Contribution

The work on this thesis can be divided in three main parts:

Rule Induction with *Artificial Neural Network*s (ANNs). Traditional ANN architectures scale well with the size of the data. But, they are defined 'black-box' models as it is difficult to understand the decisions made by the model when giving predictions. On the contrary, rule induction algorithms provide interpretable rules induced from data, but face difficulties in scaling in the presence of a high number of variables. In Chapter 3, we introduced RIDDLE. A novel ANN architecture based on the framework of possibility theory. The only requirement needed to run the learning algorithm is to express the features of the dataset used for training with possibility values. Each one of these values is associated to the truth value of the corresponding variable in the original dataset. We implemented the RIDDLE architecture with the openly available and industrial-ready

PyTorch library [Paszke et al., 2019]. The parameter optimisation task is carried with standard and established techniques based on first-order derivatives. The formalisation is easily applicable to future extension or discovery of new techniques for parameter optimisations. In the experimental evaluation with 15 real-world datasets, we compared the model complexity of RIDDLE and the fuzzy rule decision tree FURIA [Hühn and Hüllermeier, 2009]. We show that the extracted rules are most of the times less complex than rules found by the decision tree. We concluded that RIDDLE is competitive and a viable option for rule induction tasks.

Computational Learning Theory. The empirical performance of RIDDLE is promising, and we can apply it to solve challenging rule induction tasks. However, the arguments shown in Chapter 3 do not provide properties about the learned rules or the learning task itself. For this reason, in Chapter 4 we formally investigated the learning task of learning possibilistic logic theories in light of Angluin's *Exact Learning* (EL) model [Angluin, 1988]. To do so, we introduced new notions in computational learning theory, bridging the gap between the theory of computation and computational learning theory. We provided the definition of a *learning system* that abstractly models the communication between a learner and a teacher. Both of them are pictured as Turing Machines, and the learner represents the learning algorithm that identifies an unknown target hypothesis. The generality of the learning system allows to apply different learning models present in literature like the EL and *Probably Approximately Correct* (PAC) model [Valiant, 1984]. We first identify conditions that guarantee the learner to succeed in finding a possibilistic theory equivalent to the target. Then, we study conditions that allow a polynomial time learning algorithm to be adapted and used in the possibilistic setting. We consider cases where the learner can ask the queries: membership, equivalence, superset, subset only, or membership and equivalence queries together. Our results apply in the learning from entailment setting and when the classical logic language used to define the possibilistic extension is a decidable fragment of *First Order* (FO) logic. We also recall a connection between the EL and the PAC learning model. Indeed, polynomial time results in the EL model with only equivalence queries can be transferred to the PAC learning model [Angluin, 1988; Mohri et al., 2012]. If the learner is allowed to ask membership queries too, the learnability results holds in the PAC model where the learner can ask example and membership queries.

Learning Possibilistic Horn Theories. The formal study carried in Chapter 4, encouraged the development of a possibilistic version of the LRN algorithm [Frazier and Pitt, 1993b] that learns (propositional) Horn formulas in the EL model by asking membership and equivalence queries. LRN is developed in the learning from entailment setting, but we decided to implement a version that runs in the learning from partial

interpretations setting. This change is due to the fact that representing examples as partial interpretations better captures the type of data we use during training. We name the new algorithm Π_HORN*, and we prove correctness and polynomial time running properties with respect to the size of the target and number of variables into consideration. This includes also the precision, that is the highest number of digits that the target requires to express the certainty degrees of the encoded formulas.

We also address the problem of answering the queries asked by an EL algorithm. Often, this represents an obstacle for the adoption of algorithms in such a model. Our method allows an EL algorithm to receive answers to queries in an automated way from data. To do so, we leverage the connection to the EL and PAC model. The technique can be divided in two parts. In the first, we train a *Machine Learning* (ML) model that identifies patterns in data. In the second part, we use the learned model to answer queries of an EL algorithm that aims at identifying the patterns encoded in the trained model. This technique is especially useful when the trained ML model is black-box, i.e. it is not easy to extract the patterns after it has learned after the training phase. For this reason, we select ANNs as a ML model that generalises patterns from data and that answers the queries asked by Π_HORN*. With this method, an EL algorithm loses the property of outputting a hypothesis equivalent to the target, but it outputs a hypothesis that respects PAC conditions. We empirically test our technique with the Π_HORN* algorithm that by asking queries to a trained ANN model N, it outputs a possibilistic theory that is an approximation of the rules encoded in N. The empirical evaluation up until 10 000 variables showed that this is a viable solution.

7.2 Future Work

As already mentioned in Chapter 3, although the RIDDLE architecture is efficient and scales well with the size of the dataset, we can further optimise the matrix computation of RIDDLE in our implementation and speed-up both training and inference time. Moreover, the quality of the rules found with RIDDLE improves with the quality of the possibilistic values in the input dataset. A better estimation of such values contributes positively to both the complexity of the output rules and certainty degrees associated to them. For this reason, future work may focus on improving the data preprocessing. This can be done on many aspects. For instance, there are different methods of drawing possibilities distributions from imprecise data [Dubois and Prade, 1992, 2016]. Some are based on the connection between possibility distributions and uncertain probabilities; others on qualitative analysis. Another interesting project would be to find an automated technique for discretising the original dataset, before drawing possibilities values for the

final dataset given as input to RIDDLE. As a consequence, such technique could decrease the total number of final variables in the input dataset and improve the complexity of the final RIDDLE model. On top of that, the human-time saved during data preprocessing is a huge bonus.

In Chapter 4, we carried our investigation in the learning from entailment setting in the EL model. Since the set of learning frameworks learnable in polynomial time in the EL model is only a strict subset of the learning frameworks learnable in polynomial time in the PAC model, extending our study directly to the PAC model is a viable option. The proposed learning system is general enough to be applied to both *active* (like EL) and *passive* (like PAC) learning scenarios. Next steps could consider applying it to the settings of *transfer learning*, or *multiple-instance learning*. The first focuses on providing the solution of a problem from the knowledge gained in a similar problem instance, such as learning to play chess after having mastered checkers. The second studies how to learn a concept only based on the relation of other concepts. For example, learning to identify the concept 'beach', but only looking at the relation between 'water', 'sand', etc. Finally, we may also investigate learnability and polynomial time transferability results to possibilistic settings in learning from interpretations, or learning from satisfiability setting [De Raedt, 1997].

In Chapter 5, we proposed a way of simulating equivalence queries from data. Depending on the domain at hand, there may be new ways of providing answers to queries answered by EL algorithms. Additionally, we can improve the performance of the Π_HORN* algorithm, both learning and inference time. The optimisation process may take into account the adoption of already optimised solvers. Moreover, the polynomial time learnability results shown in Chapter 4 and Chapter 5, hold for decidable fragments of FO logic. Therefore, we can extend the work present in the EL literature, and develop possibilistic versions of such EL algorithms.

Appendix A

Omitted Definitions and Proofs

In this chapter, we can find omitted proofs in the main body of the document. Appendix A.1 defines the more complex FO language. Appendix A.2 provides all the omitted proofs in Chapter 4, and Appendix A.3 shows omitted proofs in Chapter 5.

A.1 First-Order Logic

An alphabet of a *First Order* (FO) logic language $\mathfrak{L}$ consists of (1) countably infinite sets R, F, V of, respectively, *relations symbols*, *functions symbols* and *variables*; (2) the set $\{\neg/1, \wedge/2, \vee/2, \rightarrow /2, \leftrightarrow /2\}$ of boolean connectives; (3) the set $\{\forall, \exists\}$ of *quantifiers* and (4) the special characters "(", ")" and ",". R, F, V are assumed to be given, function symbols with arity 0 are called *constant symbols*, $\forall$ is called *universal quantifiers* and $\exists$ is called *existential quantifier*. Recall that a *signature* of the language is a non-empty set of symbols used to formulas. The set of predicates symbols V and functions F in the signature are assumed to be known prior to learning. Let $\mathsf{X} \coloneqq \{\mathsf{x}_1, \mathsf{x}_2, \cdots\}$ be an infinite set of variables. The set $\mathsf{T}_{\mathsf{F},\mathsf{X}}$ of *terms* is the smallest set satisfying:

1. every variable $\mathsf{x} \in \mathsf{X}$ is in $\mathsf{T}_{\mathsf{F},\mathsf{X}}$.
2. if $f/n \in \mathsf{F}$ and $t_1, \cdots, t_n \in \mathsf{T}_{\mathsf{F},\mathsf{X}}$, then $f(t_1, \cdots, t_n) \in \mathsf{T}_{\mathsf{F},\mathsf{X}}$.

We omit the subscripts from $\mathsf{T}_{\mathsf{F},\mathsf{X}}$ if clear from the context. A term is *closed* if it does not contain any variables.

The set of FO logic formulas $\mathcal{L}_{\mathsf{V},\mathsf{F},\mathsf{X}}$ is defined as follows:

1. for a *predicate* $p/n \in \mathsf{V}$ and terms $t_1, \cdots, t_n \in \mathsf{T}_{\mathsf{F},\mathsf{X}}$, the string of the form $p(t_1, \cdots, t_n)$, called *atom* is in $\mathcal{L}_{\mathsf{V},\mathsf{F},\mathsf{X}}$;
2. if $\phi \in \mathcal{L}_{\mathsf{V},\mathsf{F},\mathsf{X}}$, then $\neg\phi \in \mathcal{L}_{\mathsf{V},\mathsf{F},\mathsf{X}}$;
3. if $\phi_1, \phi_2 \in \mathcal{L}_{\mathsf{V},\mathsf{F},\mathsf{X}}$, then for any connective $\circ/2 \in \{\wedge, \vee\}$, $(\phi_1 \circ \phi_2) \in \mathcal{L}_{\mathsf{V},\mathsf{F},\mathsf{X}}$;
4. if $\phi \in \mathcal{L}_{\mathsf{V},\mathsf{F},\mathsf{X}}$, then for an $\mathsf{x} \in \mathsf{X}$, we have $(\forall \mathsf{x})\phi, (\exists \mathsf{x})\phi \in \mathcal{L}_{\mathsf{V},\mathsf{F},\mathsf{X}}$.

A *first order logic interpretation* $\mathcal{I} \coloneqq (\Delta, \cdot^{\mathcal{I}})$ consists of a non-empty set Δ, called domain, and a mapping $\cdot^{\mathcal{I}}$. For every function symbol $f/n \in \mathsf{F}$, we define the mapping $f^{\mathcal{I}} : \Delta^n \to \Delta$, and for every relation $p/n \in \mathsf{R}$, we denote the object of the domain belonging to the relation $p^{\mathcal{I}} \subseteq \Delta^n$. A *variable assigment* w.r.t. $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$ is a mapping $Z : \mathsf{V} \to \Delta$. The *image* of a variable $x \in \mathsf{V}$ with respect to Z is denoted $\mathsf{x}^Z = Z(\mathsf{x})$.[1]

The *meaning* of a term $t \in \mathsf{T}$ with respect to an interpretation $\mathcal{I}$ and a variable assignment Z is denoted $t^{\mathcal{I},\mathsf{Z}}$, and it is defined recursively:

1. for every variable $\mathsf{x} \in \mathsf{X}$, $\mathsf{x}^{\mathcal{I},\mathsf{Z}} = \mathsf{x}^{\mathsf{Z}}$;
2. for every term $f(t_1, \cdots, t_n) \in \mathsf{T}$, $f(t_1, \cdots, t_n)^{\mathcal{I},\mathsf{Z}} = f^{\mathcal{I}}(t_1^{\mathcal{I},\mathsf{Z}}, \cdots, t_n^{\mathcal{I},\mathsf{Z}})$.

An interpretation $\mathcal{I}$ and a variable assignment Z assign a truth value in $\{\top, \bot\}$ to a formula $\phi \in \mathcal{L}_{\mathsf{V},\mathsf{F},\mathsf{X}}$ as follows:

1. if $\phi = p(t_1, \cdots, t_n)$, then $p(t_1, \cdots, t_n)^{\mathcal{I},Z} = \top$ iff $(t_1^{\mathcal{I},Z}, \cdots, t_n^{\mathcal{I},Z}) \in p^{\mathcal{I}}$;
2. if $\phi = (\neg\psi)$, then $(\neg\psi)^{\mathcal{I},Z} = \neg(\psi)^{\mathcal{I},Z}$;
3. if $\phi = (\phi_1 * \phi_2)$ for a binary connective $*$, then $(\phi_1 * \phi_2)^{\mathcal{I},Z} = (\phi_1^{\mathcal{I},Z} * \phi_2^{\mathcal{I},Z})$;
4. if $\phi = (\forall x)\psi$, then $((\forall x)\psi)^{\mathcal{I},Z} = \top$ iff for all $d \in \Delta$, $\psi^{\mathcal{I},\{x \to d\}Z} = \top$;
5. if $\phi = (\exists x)\psi$, then $((\exists x)\psi)^{\mathcal{I},\mathsf{Z}} = \top$ iff for some $d \in \Delta$, $\psi^{\mathcal{I},\{x \to d\}Z} = \top$.

Satisfiability of a KB follow the same notations introduced in Chapter 2.

A.2 Proofs of Chapter 4

This section is divided in two parts. The first section shows the omitted proofs concerning learnability of learning frameworks. The second section provides the omitted proofs for the polynomial time transferability results.

[1] In this work we only consider closed formulas.

A.2.1 Learnability (Section 4.2)

Theorem 4.8. *Let $\mathfrak{F}$ be a non-trivial FO learning framework. $\mathfrak{F}_\pi$ is not in* $\textsc{El}(\mathsf{MQ})$.

Proof. Let $\mathfrak{F}_\pi = (\mathcal{E}_\pi, \mathcal{L}_\pi, \mu_\pi)$ and let $\mathcal{T} \in \mathcal{L}_\pi$ be a falsifiable target (end of 2.1 in Chapter 2). Assuming there is a terminating $T(\mathcal{T})$ (Remark 4.1), for any learner L, we prove that not every path rooted in $\mathsf{T}_{L,T(\mathcal{T}),\Sigma_\mathcal{T}}$ is terminated or if there are terminates paths, then $\mathcal{H} \in (L, T(\mathcal{T}))(\Sigma_\mathcal{T})$ does not satisfy $\mathcal{H} \equiv \mathcal{T}$.

Proof strategy. The goal of the learner is to prune the search space $\mathcal{L}_\pi$ through the information acquired by the answers of membership queries. For any $n \geq 0$, we denote with S^n the set of candidate hypotheses compliant with the set of positive and negative examples that L received after having asked the n-th membership query. Initially, $S^0 = \mathcal{L}_\pi$. Consider $(\phi, \alpha) \in \mathcal{T}$ with ϕ different from a tautology, such (ϕ, α) exists because $\mathcal{T}$ is falsifiable. By definition of $\mathcal{L}_\pi$, for any valuation $\beta \in (0, 1]$ (infinitely many) there is $\mathcal{T}^\beta \in \mathcal{L}_\pi$ where $\mathcal{T}^\beta := (\mathcal{T} \cup \{(\phi, \beta)\}) \setminus \{(\phi, \alpha)\}$. So $\mathcal{L}_\pi$ (hence S^0) contains infinitely many non-equivalent hypotheses. We prove that after any membership query asked by L, the number of non-equivalent candidate hypotheses in the search space remains infinite. That is, for every $i > 0$, we always have S^i of infinite size.

Pruning the search space. Assume the set of candidate hypotheses is S^i with $i \geq 0$ and the $i+1$-th membership query is asked as input (ψ, δ). If the answer is 'yes', then $S^{i+1} = \{\mathcal{H} \in S^i \mid \mathcal{H} \models (\psi, \delta)\}$. If for some $\beta \in (0, 1]$, $\mathcal{T}^\beta \in S^i$ and $\mathcal{T}^\beta \not\models (\psi, \delta)$, then $\mathcal{T}^\beta \notin S^{i+1}$. When this happens, we have that $\mathcal{T} \models (\psi, \delta)$ and $\mathcal{T}^\beta \not\models (\psi, \delta)$. We know $\mathcal{T}^{\beta *} = \mathcal{T}^*$ and by Point 3 of Proposition 4.7, $\mathcal{T}_\delta^{\beta *} \neq \mathcal{T}_\delta^*$. By construction of $\mathcal{T}^\beta$, it follows that $\mathcal{T}_\delta^{\beta *} \subset \mathcal{T}_\delta^*$. Therefore, we deduce that $\beta < \delta \leq \alpha$. We denote by v^i_{min} the highest valuation δ present in a positive example (asked in the first i membership queries) such that there is $\mathcal{T}^\beta \in S^{i-1}$ and $\mathcal{T}^\beta \not\models (\psi, \delta)$. We set v^i_{min} to 0 if there is no such an example.

If the answer is 'no', then $S^{i+1} = \{\mathcal{H} \in S^i \mid \mathcal{H} \not\models (\psi, \delta)\}$. Similarly as before, if for some $\beta \in (0, 1]$, $\mathcal{T}^\beta \in S^i$ and $\mathcal{T}^\beta \models (\psi, \delta)$, then $\mathcal{T}^\beta \notin S^{i+1}$. In this case, we have that $\mathcal{T} \not\models (\psi, \delta)$ and $\mathcal{T}^\beta \models (\psi, \delta)$. We know $\mathcal{T}^{\beta *} = \mathcal{T}^*$ and by Point 3 of Proposition 4.7, $\mathcal{T}_\delta^{\beta *} \neq \mathcal{T}_\delta^*$. By construction $\mathcal{T}_\delta^{\beta *} \supset \mathcal{T}_\delta^*$. In this case it follows that $\beta \geq \delta > \alpha$. We denote by v^i_{max} the lowest valuation δ present in a negative example (asked in the first i membership queries) such that there is a $\mathcal{T}^\beta \in S^{i-1}$ and $\mathcal{T}^\beta \models (\psi, \delta)$. We set v^i_{max} to 1 if there is no such an example.

Non-learnability. We have $(\phi, \alpha) \in \mathcal{T}$ and that the i-th positive example may rule out some $\mathcal{T}^\beta$ with $\beta \neq \alpha$ from the set S^i with a $\beta < \alpha$ and the i-th negative example may

rule out some $\mathcal{T}^\beta$ from the set S^i with a $\beta > \alpha$. After every i-th query, we have $\mathcal{T}^\beta \in S^i$ with $v^i_{\mathsf{min}} < \beta, \alpha < v^i_{\mathsf{max}}$. Therefore, S^i will always contain infinitely many hypotheses not equivalent with each other, independently of how many membership queries have been asked. At every step in an arbitrary path in $\mathsf{T}_{L,T(\mathcal{T}),\Sigma_\mathcal{T}}$, only finitely many membership queries have been asked by L that it is not able to identify the target, as the set of candidate hypotheses is always infinite. If L terminates with a hypothesis $\mathcal{H}$ as output, L cannot guarantee that $\mathcal{H}$ is equivalent to the target. Therefore, there is no learner such that $\mathcal{H} \in (L, T(\mathcal{T}))(\Sigma_\mathcal{T})$ and $\mathcal{H} \equiv \mathcal{T}$. □

Corollary 4.12. *Let $\mathfrak{F}$ be an FO learning framework. $\mathfrak{F}$ has a terminating teacher that answers equivalence queries iff $\mathfrak{F}$ is in* $\textsc{El}(\mathsf{EQ})$.

Proof. (⇒)Every FO learning framework $\mathfrak{F}$ with a terminating teacher is learnable with only equivalence queries. Indeed, for any terminating teacher $T_\mathfrak{F}$ (Remark 4.1), there is a learning system $(L_\mathfrak{F}, T_\mathfrak{F})$ where the naive learner $L_\mathfrak{F}$ enumerates all $\mathcal{H} \in \mathcal{L}$ built using symbols from $\Sigma_\mathcal{T}$, taken as input, and asks the possible hypothesis to oracle $\mathsf{EQ}_{\mathfrak{F},\mathcal{T}}$, one by one. The learner does not know the size of the target in advance but it can estimate it to be n, ask all possible hypothesis of this size, then increase to $n+1$, and so on. Since the target is finite, eventually $L_\mathfrak{F}$ halts and outputs $\mathcal{H}$ equivalent to $\mathcal{T}$.

(⇐) If $\mathfrak{F}$ is exactly learnable with equivalence queries, the learner is guaranteed to always receive an answer from the teacher. This means that $\mathfrak{F}$ has a terminating teacher. □

Theorem 4.15. *For every non-trivial FO learning framework $\mathfrak{F}$, we have that $\mathfrak{F}_\pi$ is neither in* $\textsc{El}(\mathsf{SpQ})$ *nor in* $\textsc{El}(\mathsf{SbQ})$.

Proof. We first show the proof when the learner can ask only superset queries. Let $\mathfrak{F}_\pi = (\mathcal{E}_\pi, \mathcal{L}_\pi, \mu_\pi)$ and let $\mathcal{T} \in \mathcal{L}_\pi$ be a falsifiable target. Assuming there is a terminating T, for any learner L that can ask only superset queries, we can prove similarly as in Theorem 4.8 that the computation tree $\mathsf{T}_{L,T(\mathcal{T}),\Sigma_\mathcal{T}}$ has infinite depth.

By definition of $\mathfrak{F}_\pi$, we have that for a formula $(\phi, \alpha) \in \mathcal{T}$ and $\beta \in (0, 1]$, there is $\mathcal{K}^{\phi,\beta} = (\mathcal{T} \setminus \{(\phi, \alpha)\}) \cup \{(\phi, \beta)\}$ ($\mathcal{K}^{\phi,\alpha}$ is $\mathcal{T}$). For simplicity we omit ϕ from $\mathcal{K}^{\phi,\beta}$ for the rest of the proof. There are infinitely many such $\mathcal{K}^\beta \in \mathcal{L}_\pi$. The proof can continue as in Theorem 4.8. Alternatively, we can assume that in a step in an arbitrary path p in $\mathsf{T}_{L,T(\mathcal{T}),\Sigma_\mathcal{T}}$ the learner knows that for $\beta \geq \alpha$, $\mu_\pi(\mathcal{T}) \subseteq \mu_\pi(\mathcal{K}^\beta)$ and for some $\gamma < \alpha$, $\mu_\pi(\mathcal{K}^\gamma) \subset \mu_\pi(\mathcal{T})$. We show that also with this assumption, the learner cannot identify the target among the candidate hypotheses $\mathcal{K}^\delta$ with $\delta \in (\gamma, \beta]$ with a finite amount of calls to the oracle.

Infinite candidate hypotheses. Let $(\gamma, \beta]$ be the interval containing all valuations δ

of candidate hypotheses $\mathcal{K}^\delta$. Whenever the learner calls $\mathsf{SpQ}_{\mathfrak{F}_\pi,\mathcal{T}}$ in p with input $\mathcal{K}^\delta$ and $\delta \in (\gamma, \beta]$, the answer 'yes' will bound the right member of the interval $(\gamma, \beta]$. That is, β will be updated with the value $\delta \leq \beta$. This is because if $\mu_\pi(\mathcal{T}) \subseteq \mu_\pi(\mathcal{K}^\delta)$, by construction $\delta \geq \alpha$. The answer 'no' will update γ with $\delta > \gamma$ because if $\mu_\pi(\mathcal{K}^\delta) \subset \mu_\pi(\mathcal{T})$, by construction $\delta < \alpha$. In both cases, the number of candidate hypotheses remains infinite. If the input to the superset query is $\mathcal{K}^\delta$ with $\delta \notin (\gamma, \beta]$, the search space of candidate hypotheses is not pruned after receiving the answer.

Non-termination. Only finitely many queries have been asked by L at every step in an arbitrary path in $\mathsf{T}_{L,T(\mathcal{T}),\Sigma_\mathcal{T}}$. Therefore, L is not able to identify the target as the set of candidate hypotheses is always infinite. Since there is no finite path in $\mathsf{T}_{L,T(\mathcal{T}),\Sigma_\mathcal{T}}$ where L terminates with a hypothesis equivalent to the target, the depth of $\mathsf{T}_{L,T(\mathcal{T}),\Sigma_\mathcal{T}}$ is infinite and $\mathfrak{F}_\pi$ is not learnable.

The case with subset queries. The proof for showing non learnability with only subset queries is similar. Let $\mathcal{T}$ be the target. We can assume that in a step in an arbitrary path p in $\mathsf{T}_{L,T(\mathcal{T}),\Sigma_\mathcal{T}}$ the learner knows that for $\beta \leq \alpha$, $\mu_\pi(\mathcal{K}^\beta) \subseteq \mu_\pi(\mathcal{T})$ and for some $\gamma > \alpha$, $\mu_\pi(\mathcal{T}) \subset \mu_\pi(\mathcal{K}^\gamma)$. The learner will search for a target $\mathcal{K}^\delta$ in the interval $\delta \in [\beta, \gamma)$ by asking subset queries. Upon asking a query with input (ϕ, η), the answer 'yes' will bound the left argument β of the interval of candidate valuations with a $\eta \geq \beta$. This is because if $\mu_\pi(\mathcal{K}^\eta) \subseteq \mu_\pi(\mathcal{T})$, by construction $\eta \leq \alpha$. The answer 'no' will update γ with $\eta < \gamma$ because if $\mu_\pi(\mathcal{T}) \subset \mu_\pi(\mathcal{K}^\eta)$, by construction $\eta > \alpha$. As proved in the first part, the number of candidate hypotheses remains infinite at every step in any path of the computation tree. □

Theorem 4.16. *Let $\mathfrak{F} = (\mathcal{E}, \mathcal{L}, \mu)$ be a safe FO learning framework and for any $p \in \mathbb{N}^+$ let $\mathfrak{F}^p_\pi = (\mathcal{E}_\pi, \mathcal{L}^p_\pi, \mu)$ be its possibilistic extension. $\mathfrak{F}$ is in* $\textsc{El}(\mathsf{SbQ})$ *iff $\mathfrak{F}^p_\pi$ is in* $\textsc{El}(\mathsf{SbQ})$.

Proof. ($\Rightarrow$) Let $\mathcal{T} \in \mathcal{L}^p_\pi$ be the target, and let $T_{\mathfrak{F}_\pi}$ be a terminating teacher, we describe the action of a learner $L_{\mathfrak{F}_\pi}$ such that the computation tree of $(L_{\mathfrak{F}_\pi}, T_{\mathfrak{F}_\pi}(\mathcal{T}))$ with input $\Sigma_\mathcal{T}$ has finite depth and $\mathcal{H} \in (L_{\mathfrak{F}_\pi}, T_{\mathfrak{F}_\pi}(\mathcal{T}))(\Sigma_\mathcal{T})$ satisfies $\mathcal{H} \equiv \mathcal{T}$. $\mathfrak{F}$ in $\textsc{ELP}(\mathsf{SbQ})$ implies that there is a learning system $(L_\mathfrak{F}, T_\mathfrak{F})$ such that for any $\mathcal{K} \in \mathcal{L}$, $\mathsf{T}_{L_\mathfrak{F}, T_\mathfrak{F}(\mathcal{K}), \Sigma_k}$ has a finite depth, $L_\mathfrak{F}$ calls only $\mathsf{SbQ}_{\mathfrak{F},\mathcal{K}}$, and $\mathcal{H}' \in (L_\mathfrak{F}, T_\mathfrak{F})(\Sigma_\mathcal{K})$ implies $\mathcal{H}' \equiv \mathcal{K}$. Since $\mathfrak{F}$ is safe, we have that $\mathcal{T}^*_\alpha \in \mathcal{L}$ for any $\alpha \in (0, 1]_p$. We are going to use the following claim.

Claim A.1. *Let $\mathfrak{F} = (\mathcal{E}, \mathcal{L}, \mu)$ be a safe FO learning framework and let $\mathfrak{F}_\pi = (\mathcal{E}_\pi, \mathcal{L}_\pi, \mu_\pi)$ be its possibilistic extension. Let $\mathcal{T} \in \mathcal{L}_\pi$, and $\mathcal{K} \in \mathcal{L}$. For a fixed $\alpha \in (0, 1]$, let $\mathcal{H} = \{(\phi, \alpha) \mid \phi \in \mathcal{K}\}$. The oracle $\mathsf{SbQ}_{\mathfrak{F},\mathcal{T}^*_\alpha}(\mathcal{K})$ can return the answer ψ iff $\mathsf{SbQ}_{\mathfrak{F}_\pi,\mathcal{T}}(\mathcal{H})$ can return (ψ, β) with $\beta \leq \alpha$.*

Proof. Assume $\mathsf{SbQ}_{\mathfrak{F},\mathcal{T}^*_\alpha}(\mathcal{K})$ returned ψ, hence $\mathcal{K} \models \psi$ and $\mathcal{T}^*_\alpha \not\models \psi$. By construction, $\mathcal{K} = \mathcal{H}^*_\alpha \models \psi$ and by Point 3 of Proposition 4.7, $\mathcal{T} \not\models (\psi, \alpha)$ and $\mathcal{H} \models (\psi, \alpha)$. Therefore $\mathsf{SbQ}_{\mathfrak{F}_\pi,\mathcal{T}}(\mathcal{H})$ can return (ψ, β) with $\beta \leq \alpha$. Assume that $\mathsf{SbQ}_{\mathfrak{F}_\pi,\mathcal{T}}(\mathcal{H})$ returned (ψ, β) with $\beta \leq \alpha$. This means that $\mathcal{H} \models (\psi, \beta)$ and $\mathcal{T} \not\models (\psi, \beta)$ (with $\beta > 0$). By Point 5 of Proposition 4.7 and by construction of $\mathcal{H}$, $\mathsf{val}(\phi, \mathcal{H}) = \alpha$, then $\mathcal{H} \models (\psi, \alpha)$. By Point 3 of Proposition 4.7, $\mathcal{H}^*_\alpha \models \psi$ and since $\mathcal{H}^*_\alpha = \mathcal{K}$, we have that $\mathcal{K} \models \psi$. Finally, by Proposition 2.8 we get $\mathcal{T}^*_\alpha \not\models \psi$. □

Description of $L_{\mathfrak{F}_\pi}$'s steps. For each $\alpha \in (0, 1]_p$, the learner $L_{\mathfrak{F}_\pi}$ repeats the same steps performed by the learner $L_{\mathfrak{F}}$ in $\mathsf{T}_{L_{\mathfrak{F}},T_{\mathfrak{F}}(\mathcal{T}^*_\alpha),\Sigma_{\mathcal{T}^*}}$. At the i-th call to $\mathsf{SbQ}_{\mathfrak{F},\mathcal{T}^*_\alpha}$ with input $\mathcal{K}^\alpha$ in p in $\mathsf{T}_{L_{\mathfrak{F}},T_{\mathfrak{F}}(\mathcal{T}^*_\alpha),\Sigma_{\mathcal{T}^*}}$, the learner $L_{\mathfrak{F}_\pi}$ calls $\mathsf{SbQ}_{\mathfrak{F}_\pi,\mathcal{T}}$ with $\mathcal{H}^\alpha = \{(\phi, \alpha) \mid \phi \in \mathcal{K}^\alpha\}$ as input. This hypothesis is created in finitely many steps.

By Claim A.1, $L_{\mathfrak{F}_\pi}$ is able to perform every step made by $L_{\mathfrak{F}}$ in p. Upon receiving a counterexample ϕ, it continues the steps made by $L_{\mathfrak{F}}$ as the subset query returned (ϕ, α). Since $\mathsf{T}_{L,T(\mathcal{K}),\Sigma_\mathcal{K}}$ is finite and $(0, 1]_p$ contains finitely many valuations, $L_{\mathfrak{F}_\pi}$ will be able to find a $\mathcal{K}^*_\alpha \equiv \mathcal{T}^*_\alpha$ for each $\alpha \in (0, 1]_p$ in finitely many steps. By Lemma 4.9, $\mathcal{T} \equiv \mathcal{H} = \{(\phi, \alpha) \mid \phi \in \mathcal{K}^*_\alpha, \alpha \in (0, 1]_p\}$.

Termination. Let d be the longest (finite) depth of $\mathsf{T}_{L_{\mathfrak{F}},T_{\mathfrak{F}}(\mathcal{T}^*_\alpha),\Sigma_{\mathcal{T}^*}}$ for $\alpha \in (0, 1]_p$. The depth of the computation tree of $(L_{\mathfrak{F}_\pi}, T_{\mathfrak{F}_\pi}(\mathcal{T}))$ with input $\Sigma_\mathcal{T}$ is bounded by d times the number of values in $(0, 1]_p$ plus a constant factor that comprises the computation needed to rewrite queries asked by $L_{\mathfrak{F}}$ and the final computation of $\mathcal{H} \equiv \mathcal{T}$. Thus, we can transfer learnability of the learning framework $\mathfrak{F}$ with only subset queries to the respective possibilistic extension $\mathfrak{F}^p_\pi$.

($\Leftarrow$) We now show the other direction. Let $\mathcal{K} \in \mathcal{L}$ be the target, and assume $T_{\mathfrak{F}}(\mathcal{K})$ is a terminating teacher. We describe the action of a learner $L_{\mathfrak{F}}$ that asks only subset queries and such that $\mathsf{T}_{L_{\mathfrak{F}},T_{\mathfrak{F}}(\mathcal{K}),\Sigma_\mathcal{K}}$ has a finite depth and that $\mathcal{H}' \in (L_{\mathfrak{F}}, T_{\mathfrak{F}}(\mathcal{K}))(\Sigma_\mathcal{K})$ satisfies $\mathcal{H}' \equiv \mathcal{K}$. We know $\mathfrak{F}^p_\pi$ is learnable, therefore there is a terminating teacher $T_{\mathfrak{F}^p_\pi}$ and a learner $L_{\mathfrak{F}^p_\pi}$ for $\mathfrak{F}^p_\pi$ that asks only subset queries, such that for every $t \in \mathcal{L}_\pi$, $\mathsf{T}_{L_{\mathfrak{F}^p_\pi},T_{\mathfrak{F}^p_\pi}(\mathcal{T}),\Sigma_\mathcal{T}}$ is finite and $\mathcal{H} \in (L_{\mathfrak{F}^p_\pi}, T_{\mathfrak{F}^p_\pi}(\mathcal{T}))(\Sigma_\mathcal{T})$ implies $\mathcal{H} \equiv \mathcal{T}$.

Description of $L_{\mathfrak{F}}$'s steps. By definition of $\mathfrak{F}^p_\pi$, we have that $\mathcal{T} = \{(\phi, 1) \mid \phi \in \mathcal{K}\} \in \mathcal{L}_\pi$. $L_{\mathfrak{F}}$ repeats every step that $L_{\mathfrak{F}^p_\pi}$ performs in an arbitrary path p in $\mathsf{T}_{L_{\mathfrak{F}^p_\pi},T_{\mathfrak{F}^p_\pi}(\mathcal{T}),\Sigma_\mathcal{T}}$. At the i-th call to $\mathsf{SbQ}_{\mathfrak{F}^p_\pi,\mathcal{T}}$ with input $\mathcal{H}$ in p, $L_{\mathfrak{F}}$ calls $\mathsf{SbQ}_{\mathfrak{F},\mathcal{K}}$ with $\mathcal{H}^*$ as input. By Claim A.1, upon receiving a counterexample ϕ, we know that $\mathcal{K} \not\models \phi$ and $\mathcal{H}^* \models \phi$. By construction of $\mathcal{T}$, $\mathcal{K} = \mathcal{T}^*_1 \not\models \phi$ and $\mathcal{T} \not\models (\phi, \alpha)$ for every $\alpha \in (0, 1]_p$. Therefore, $L_{\mathfrak{F}}$ returns $(\phi, 10^{-p})$ to $L_{\mathfrak{F}^p_\pi}$.

Termination. Each query can be translated in polynomial time w.r.t. the size of $\mathcal{H}$. Since $\mathsf{T}_{L_{\mathfrak{F}^p_\pi}, T_{\mathfrak{F}^p_\pi}(\mathcal{T}), \Sigma_\mathcal{T}}$ is finite, also $\mathsf{T}_{L_\mathfrak{F}, T_\mathfrak{F}(\mathcal{K}), \Sigma_\mathcal{K}}$ is finite. Thus, we can transfer learnability of $\mathfrak{F}^p_\pi$ (with only subset queries) to $\mathfrak{F}$. □

Theorem 4.17. *Let $\mathfrak{F} = (\mathcal{E}, \mathcal{L}, \mu)$ be a safe FO learning framework and, for any $p \in \mathbb{N}^+$, let $\mathfrak{F}^p_\pi = (\mathcal{E}_\pi, \mathcal{L}^p_\pi, \mu)$ be its possibilistic extension. $\mathfrak{F}$ is in* $\textsc{El}(\mathsf{SpQ})$ *iff $\mathfrak{F}^p_\pi$ is in* $\textsc{El}(\mathsf{SpQ})$.

Proof. ($\Rightarrow$) We show first that if $\mathfrak{F}$ is learnable, then $\mathfrak{F}^p_\pi$ is learnable. Let $\mathcal{T} \in \mathcal{L}^p_\pi$ be the target, and let $T_{\mathfrak{F}^p_\pi}$ be a terminating teacher, we describe the action of a learner $L_{\mathfrak{F}^p_\pi}$ such that the computation tree of $(L_{\mathfrak{F}^p_\pi}, T_{\mathfrak{F}^p_\pi}(\mathcal{T}))$ with input $\Sigma_\mathcal{T}$ has finite depth and $\mathcal{H} \in (L_{\mathfrak{F}^p_\pi}, T_{\mathfrak{F}^p_\pi}(\mathcal{T}))(\Sigma_\mathcal{T})$ satisfies $\mathcal{H} \equiv \mathcal{T}$. $\mathfrak{F}$ in $\textsc{ElP}(\mathsf{SpQ})$ implies that there is a learning system $(L_\mathfrak{F}, T_\mathfrak{F})$ such that for any $\mathcal{K} \in \mathcal{L}$, $\mathsf{T}_{L_\mathfrak{F}, T_\mathfrak{F}(\mathcal{K}), \Sigma_\mathcal{K}}$ has a finite depth, and $\mathcal{H}' \in (L_\mathfrak{F}, T_\mathfrak{F})(\Sigma_\mathcal{K})$ implies $\mathcal{H}' \equiv \mathcal{K}$. Since $\mathfrak{F}$ is safe, we have that $\mathcal{T}^*_\alpha \in \mathcal{L}$ for any $\alpha \in (0, 1]_p$.

Description of $L_{\mathfrak{F}_\pi}$'s steps. For each $\alpha \in (0, 1]_p$, the learner $L_{\mathfrak{F}^p_\pi}$ repeats the same steps performed by the learner $L_\mathfrak{F}$ in an arbitrary path p_α in $\mathsf{T}_{L_\mathfrak{F}, T_\mathfrak{F}(\mathcal{T}^*_\alpha), \Sigma_{\mathcal{T}^*}}$. When in p_α there is a call to $\mathsf{SpQ}_{\mathfrak{F}, \mathcal{T}^*_\alpha}$ with input $\mathcal{K}^\alpha$, $L_{\mathfrak{F}^p_\pi}$ does not ask any query (it temporarily stops the simulation of $L_\mathfrak{F}$'s steps at the point where $L_\mathfrak{F}$ stops in superset query state). When $L_{\mathfrak{F}^p_\pi}$ reaches the point where $L_\mathfrak{F}$ stops in superset query state with input $\mathcal{K}^\beta$ for every $\beta \in (0, 1]_p$, it calls $\mathsf{SpQ}_{\mathfrak{F}_\pi, \mathcal{T}}$ with

$$\mathcal{H}^\alpha = \bigcup_{\beta \in (0,1]_p} \{(\phi, \alpha) \mid \phi \in \mathcal{K}^\beta\}$$

as input. This hypothesis is created in finitely many steps.

Claim A.2. *Let $\mathfrak{F} = (\mathcal{E}, \mathcal{L}, \mu)$ be a safe FO learning framework and let $\mathfrak{F}^p_\pi = (\mathcal{E}_\pi, \mathcal{L}^p_\pi, \mu_\pi)$ be its possibilistic extension. Let $\mathcal{T}, \mathcal{H} \in \mathcal{L}^p_\pi$, if $\mathsf{SpQ}_{\mathfrak{F}^p_\pi, \mathcal{T}}(\mathcal{H})$ can return (ψ, β), then $\mathsf{SpQ}_{\mathfrak{F}, \mathcal{T}^*_\beta}(\mathcal{H}^*_\beta)$ can return ψ.*

Proof. Assume a counterexample (ϕ, β) is returned after the call to $\mathsf{SpQ}_{\mathfrak{F}^p_\pi, \mathcal{T}}$ with input $\mathcal{H}$. We have that $\mathcal{H} \not\models (\phi, \beta)$ and $\mathcal{T} \models (\phi, \beta)$. By Point 3 of Proposition 4.7, we have that $\mathcal{T}^*_\beta \models \phi$ and $\mathcal{H}^*_\beta \not\models \phi$. □

Assume a counterexample (ϕ, α) is returned after the call to $\mathsf{SpQ}_{\mathfrak{F}_\pi, \mathcal{T}}$ with input $\mathcal{H}$. By Claim A.2 and by construction of $\mathcal{H}$, we know that there is a $\mathcal{K}^\beta$ such that $\mathcal{K}^\beta \not\models \phi$ for $\beta \in (\alpha, 1]_p$. Therefore, in this way $L_{\mathfrak{F}^p_\pi}$ finds a counterexample for at least one $\mathcal{K}^\alpha$ and we may use it to resume the computation in path p_α. With this consideration, upon receiving a counterexample (ψ, γ), $L_{\mathfrak{F}^p_\pi}$ loops for every $\beta \in (0, \gamma]_p$ and if $\mathcal{K}^\beta \not\models \psi$, it

continues to perform the steps made by $L_{\mathfrak{F}}$ in p_β treating ψ as a counterexample. When for all $\delta \in (0,1]_p$, $L_{\mathfrak{F}_\pi^p}$ reached the point where p_δ is waiting in query state with input $\mathcal{K}^\delta$, it generates again $\mathcal{H}$ as defined until the superset query returns 'yes'.

Termination. $L_{\mathfrak{F}_\pi^p}$ is able to perform every step made by $L_{\mathfrak{F}}$ in p. Since $\mathsf{T}_{L_{\mathfrak{F}}, T_{\mathfrak{F}}(\mathcal{K}), \Sigma_{\mathcal{K}}}$ is finite and $(0,1]_p$ contains finitely many valuations, $L_{\mathfrak{F}_\pi^p}$ will be able to find a $\mathcal{K}_\alpha^* \equiv \mathcal{T}_\alpha^*$ for each $\alpha \in (0,1]_p$ in finitely many steps. By Lemma 4.9,

$$\mathcal{T} \equiv \mathcal{H} = \{(\phi, \alpha) \mid \phi \in \mathcal{K}_\alpha^*, \alpha \in (0,1]_p\}.$$

Let d be the longest (finite) depth of $\mathsf{T}_{L_{\mathfrak{F}}, T_{\mathfrak{F}}(\mathcal{T}_\alpha^*), \Sigma_{\mathcal{T}^*}}$ for every $\alpha \in (0,1]_p$. The depth of the computation tree of $(L_{\mathfrak{F}_\pi^p}, T_{\mathfrak{F}_\pi^p}(\mathcal{T}))$ with input $\Sigma_{\mathcal{T}}$ is bounded by d times the number of values in $(0,1]_p$ (which is polynomial w.r.t. p) plus a constant factor that comprises the computation needed to rewrite queries asked by $L_{\mathfrak{F}}$ and the final computation of $\mathcal{H} \equiv \mathcal{T}$. Thus, we can transfer learnability of the learning framework $\mathfrak{F}$ with only superset queries to its possibilistic extension $\mathfrak{F}_\pi^p$.

($\Leftarrow$) We now show the other direction. Let $\mathcal{K} \in \mathcal{L}$ be the target, and assume $T_{\mathfrak{F}}(\mathcal{K})$ is a terminating teacher. We describe the action of a learner $L_{\mathfrak{F}}$ that asks only subset queries and such that $\mathsf{T}_{L_{\mathfrak{F}}, T_{\mathfrak{F}}(\mathcal{K}), \Sigma_{\mathcal{K}}}$ has a finite depth and that $\mathcal{H}' \in (L_{\mathfrak{F}}, T_{\mathfrak{F}}(\mathcal{K}))(\Sigma_{\mathcal{K}})$ satisfies $\mathcal{H}' \equiv \mathcal{K}$. We know $\mathfrak{F}_\pi^p$ is learnable, therefore there is a terminating teacher $T_{\mathfrak{F}_\pi^p}$ and a learner $L_{\mathfrak{F}_\mathcal{T}}$ for $\mathfrak{F}_\pi^p$ that asks only subset queries, such that for every $t \in \mathcal{L}_\pi$, $\mathsf{T}_{L_{\mathfrak{F}_\pi^p}, T_{\mathfrak{F}_\pi^p}(\mathcal{T}), \Sigma_{\mathcal{T}}}$ is finite and $\mathcal{H} \in (L_{\mathfrak{F}_\pi^p}, T_{\mathfrak{F}_\pi^p}(\mathcal{T}))(\Sigma_{\mathcal{T}})$ implies $\mathcal{H} \equiv \mathcal{T}$. By definition of $\mathfrak{F}_\pi^p$, we have that

$$\mathcal{T} = \{(\phi, 1) \mid \phi \in \mathcal{K}\} \in \mathcal{L}_\pi.$$

Description of $L_{\mathfrak{F}}$'s steps. $L_{\mathfrak{F}_\pi^p}$ repeats the steps performed in an arbitrary path p in $\mathsf{T}_{L_{\mathfrak{F}_\pi^p}, T_{\mathfrak{F}_\pi^p}(\mathcal{T}), \Sigma_{\mathcal{T}}}$. At the i-th call to $\mathsf{SpQ}_{\mathfrak{F}_\pi^p, \mathcal{T}}$ with input $\mathcal{H}$ in p, $L_{\mathfrak{F}}$ calls $\mathsf{SpQ}_{\mathfrak{F}, \mathcal{K}}$ with $\mathcal{H}^*$ as input. Upon receiving a counterexample ϕ, we know that $\mathcal{K} \models \phi$ and $\mathcal{H}^* \not\models \phi$. Consequently, by Point 3 of Proposition 4.7, $\mathcal{H} \not\models (\phi, \alpha)$ for every $\alpha \in (0,1]$. By construction of $\mathcal{T}$, $\mathcal{K} = \mathcal{T}_1^* \models \phi$ and $\mathcal{T} \models (\phi, \alpha)$ for every $\alpha \in [0,1]$. Therefore, $L_{\mathfrak{F}}$ returns $(\phi, 1)$ to $L_{\mathfrak{F}_\pi^p}$. When $L_{\mathfrak{F}_\pi}$ terminates with output $\mathcal{H}$ s.t. $\mathcal{H} \equiv \mathcal{T}$, $L_{\mathfrak{F}}$ outputs $\mathcal{H}^*$ that satisfies $\mathcal{H}^* \equiv \mathcal{T}^* = \mathcal{K}$.

Termination. Each query asked by $L_{\mathfrak{F}_\pi}$ can be translated in polynomial time w.r.t. the size of $\mathcal{H}$. Let d be the depth of $\mathsf{T}_{L_{\mathfrak{F}}, T_{\mathfrak{F}}(\mathcal{K}), \Sigma_{\mathcal{K}}}$ (which is finite). The depth of $\mathsf{T}_{L_{\mathfrak{F}_\pi^p}, T_{\mathfrak{F}_\pi^p}(\mathcal{T}), \Sigma_{\mathcal{T}}}$ is at most d plus a constant amount steps needed to simulate $\mathsf{SpQ}_{\mathfrak{F}_\pi, \mathcal{T}}$ for each superset query asked by $L_{\mathfrak{F}_\pi}$, hence it is finite. Thus, we can transfer learnability of $\mathfrak{F}_\pi^p$ (with only subset queries) to $\mathfrak{F}$. □

A.2.2 Polynomial Time Reduction (Section 4.3)

Theorem 4.19. *Let $\mathfrak{F}$ be an FO learning framework. If its possibilistic extension $\mathfrak{F}_\pi$ is in* ELP(MQ,EQ)*, then $\mathfrak{F}$ is in* ELP(MQ,EQ)*.*

Proof. Let $\mathfrak{F} = (\mathcal{E}, \mathcal{L}, \mu)$, let $\mathcal{K} \in \mathcal{L}$ be the target and $T_\mathfrak{F}$ be a terminating teacher. We list the (polynomially many) steps that a learner $L_\mathfrak{F}$ does, such that $\mathsf{T}_{L_\mathfrak{F}, T_\mathfrak{F}(\mathcal{K}), \Sigma_\mathcal{K}}$ is finite, and $\mathcal{K}' \in (L_\mathfrak{F}, T_\mathfrak{F}(\mathcal{K}))(\Sigma_\mathcal{K})$ implies $\mathcal{K}' \equiv \mathcal{K}$.

Since $\mathfrak{F}_\pi$ is in ELP(MQ,EQ), for any $\mathcal{T} \in \mathcal{L}_\pi$ there is a learning system $(L_{\mathfrak{F}_\pi}, T_{\mathfrak{F}_\pi}(\mathcal{T}))$ such that $\mathsf{T}_{L_{\mathfrak{F}_\pi}, T_{\mathfrak{F}_\pi}(\mathcal{T}), \Sigma_\mathcal{T}}$ has finite depth and $\mathcal{H} \in (L_{\mathfrak{F}_\pi}, T_{\mathfrak{F}_\pi}(\mathcal{T}))(\Sigma_\mathcal{T})$ implies $\mathcal{H} \equiv \mathcal{T}$. By definition of $\mathfrak{F}_\pi$, we have that $\mathcal{T} = \{(\phi, 1) \mid \phi \in \mathcal{K}\} \in \mathcal{L}_\pi$. We can assume that $L_\mathfrak{F}$ repeats every step that $L_{\mathfrak{F}_\pi}$ performs in an arbitrary path p in $\mathsf{T}_{L_{\mathfrak{F}_\pi}, T_{\mathfrak{F}_\pi}(\mathcal{T}), \Sigma_\mathcal{T}}$ (the signature $\Sigma_\mathcal{T}$ is built based on the symbols in $\Sigma_\mathcal{K}$).

At the i-th call to $\mathsf{MQ}_{\mathfrak{F}_\pi, \mathcal{T}}$ with input (ϕ, α) in p, $L_\mathfrak{F}$ calls $\mathsf{MQ}_{\mathfrak{F}, \mathcal{K}}$ with input ϕ. By Lemma 4.10, for all $\alpha \in (0, 1]$, $\mathsf{MQ}_{\mathfrak{F}_\pi, \mathcal{T}}((\phi, \alpha)) = \mathsf{MQ}_{\mathfrak{F}, \mathcal{K}}(\phi)$. At the i-th call to $\mathsf{EQ}_{\mathfrak{F}_\pi, \mathcal{T}}$ with $\mathcal{H}$ as input in p, $L_\mathfrak{F}$ calls $\mathsf{EQ}_{\mathfrak{F}, \mathcal{K}}$ with input $\mathcal{H}^*$. By Remark 4.3, we can assume that all counterexamples returned by $\mathsf{EQ}_{\mathfrak{F}, \mathcal{K}}$ are positive. Again by Lemma 4.10, for all $\alpha \in (0, 1]$, $\mathcal{K} \models \phi$ iff $t \models (\phi, \alpha)$, in particular, for $\alpha = 1$. This means that whenever $L_\mathfrak{F}$ receives a (positive) counterexample ϕ, it is like $(\phi, 1)$ is returned by $\mathsf{EQ}_{\mathfrak{F}_\pi, \mathcal{T}}$. Each translation of a query can be computed in polynomial time by $L_\mathfrak{F}$ w.r.t. its size. Eventually, $L_\mathfrak{F}$ computes $\mathcal{H} \equiv \mathcal{T}$. Clearly, $\mathcal{H}^* \equiv \mathcal{T}^* = \mathcal{K}$ is as required. The number of steps made by $L_\mathfrak{F}$, which define the length of paths in $\mathsf{T}_{L_\mathfrak{F}, T_\mathfrak{F}(\mathcal{K}), \Sigma_\mathcal{K}}$ is polynomial w.r.t. the sum of steps made by $L_{\mathfrak{F}_\pi}$ in p which in turn is polynomial w.r.t. $|\mathcal{T}|$ and the longest counterexample seen so far. As $|\mathcal{T}|$ is polynomial w.r.t. $|\mathcal{K}|$, by definition we have that $\mathfrak{F}$ is in ELP(MQ,EQ). □

Theorem 4.35. *Let $\mathfrak{F}_\pi$ be the possibilistic extension of an FO learning framework $\mathfrak{F}$. If $\mathfrak{F}_\pi$ in* ELP(EQ)*, then $\mathfrak{F}$ is in* ELP(EQ)*.*

Proof. let $\mathfrak{F} = (\mathcal{E}, \mathcal{L}, \mu)$ be a learning framework and assume $\mathfrak{F}_\pi = (\mathcal{E}_\pi, \mathcal{L}_\pi, \mu_\pi)$ is in ELP(EQ). Since $\mathfrak{F}_\pi$ is in ELP(EQ), there is a learning system $(L_{\mathfrak{F}_\pi}, T_{\mathfrak{F}_\pi})$ such that for any $\mathcal{T} \in \mathcal{L}_\pi$, $\mathsf{T}_{L_{\mathfrak{F}_\pi}, T_{\mathfrak{F}_\pi}(\mathcal{T}), \Sigma_\mathcal{T}}$ has a depth which is polynomial with respect to $|\mathcal{T}|$ and the largest counterexample received in each path, and $\mathcal{K}' \in (L_{\mathfrak{F}_\pi}, T_{\mathfrak{F}_\pi})(\Sigma_\mathcal{T})$ implies $\mathcal{K}' \equiv \mathcal{T}$. For a fixed but arbitrary $\mathcal{K} \in \mathcal{L}$, we consider the computation of the learning system $(L_\mathfrak{F}, T_\mathfrak{F}(\mathcal{K}))$ on input $\Sigma_\mathcal{K}$ where $T_\mathfrak{F}(\mathcal{K})$ is terminating.

Description of $L_\mathfrak{F}$'s steps. By definition of $\mathfrak{F}_\pi$, we have that $\mathcal{T} = \{(\phi, 1) \mid \phi \in \mathcal{K}\} \in \mathcal{L}_\pi$ (by definition $\Sigma_\mathcal{T}$ and $\Sigma_\mathcal{K}$ are equal). Whenever in p there is a call to $\mathsf{EQ}_{\mathfrak{F}_\pi, \mathcal{T}}$ with input $\mathcal{H}$, $L_\mathfrak{F}$ calls $\mathsf{EQ}_{\mathfrak{F}, \mathcal{K}}$ with input $\mathcal{H}^*$. Upon receiving a positive counterexample ϕ,

$L_{\mathfrak{F}}$ can continue the computation that $L_{\mathfrak{F}_\pi}$ performs with the positive counterexample $(\phi, 1) \in \mathcal{E}_\pi$. If ϕ is a negative counterexample, $(\phi, \mathsf{val}(\phi, \mathcal{H}))$ is returned instead.

Simulating a positive counterexample. Upon receiving a positive counterexample ϕ, we know that $\mathcal{K} \models \phi$ and $\mathcal{H}^* \not\models \phi$. By construction we have $\mathcal{T}_1^* = \mathcal{K}$, and it follows that $\mathcal{T}_1^* \models \phi$ iff $\mathcal{K} \models \phi$. By Point 3 of Proposition 4.7, $\mathcal{T}_1^* \models \phi$ iff $\mathcal{T} \models (\phi, 1)$, therefore $L_{\mathfrak{F}}$ can continue the computation assuming that it has received the positive counterexample $(\phi, 1) \in \mathcal{E}_\pi$.

Simulating a negative counterexample. Upon receiving a negative counterexample ϕ, we know that $\mathcal{K} \not\models \phi$ and $\mathcal{H}^* \models \phi$. Let $\alpha \in \mathcal{H}^v$ be the smallest valuation in $\mathcal{H}^v$. By construction, we get $\mathcal{H}^* = \mathcal{H}^*_\alpha$, so $\mathcal{H}^*_\alpha \models \phi$. By Point 3 of Proposition 4.7, we have $\mathcal{H}^*_\alpha \models \phi$ iff $\mathcal{H} \models (\phi, \alpha)$ and by Point 5 of Proposition 4.7, the fact $\mathcal{H} \models (\phi, \alpha)$ implies $\mathsf{val}(\phi, \mathcal{H}) \in \mathcal{H}^v \cup \{1\}$. It follows that necessarily $0 < \mathsf{val}(\phi, \mathcal{H})$. Also $\mathcal{K} \not\models \phi$ and by construction, $\mathcal{T} \not\models (\phi, 1)$ holds. By Point 4 of Proposition 4.7, $\mathsf{val}(\phi, \mathcal{T}) < 1$. Assume $\mathsf{val}(\phi, \mathcal{T}) > 0$. By Points 4 and 5 of Proposition 4.7, $1 \neq \mathsf{val}(\phi, \mathcal{T}) \in \{1\} = \mathcal{T}^v$, clearly impossible. Therefore $\mathsf{val}(\phi, \mathcal{T}) = 0$. We have shown $\mathsf{val}(\phi, \mathcal{H}) > 0$ and $\mathsf{val}(\phi, \mathcal{T}) = 0$, as a consequence $L_{\mathfrak{F}_\pi}$ can assume it has received the negative counterexample $(\phi, \mathsf{val}(\phi, \mathcal{H}))$.

Termination. By the depth of $\mathsf{T}_{L_{\mathfrak{F}_\pi}, T_{\mathfrak{F}_\pi}(\mathcal{T}), \Sigma_\mathcal{T}}$, $L_{\mathfrak{F}_\pi}$ terminates in polynomial time w.r.t. $|\mathcal{T}|$ and the largest counterexample seen so far and outputs a hypothesis $\mathcal{H}$ that satisfies $\mathcal{H} \equiv \mathcal{T}$. As $\mathcal{H}^* \equiv \mathcal{T}^* = \mathcal{K}$, $L_{\mathfrak{F}}$ finds an equivalent hypothesis $\mathcal{H}*$ w.r.t. the target $\mathcal{K}$. As a consequence we have found a polynomial time learner for $\mathfrak{F}$ and we have that $\mathfrak{F}$ is in ELP(EQ) (translating counterexamples as described can be done in polynomial time w.r.t. the size of largest counterexample received so far). □

Theorem 4.42. *Let $\mathfrak{F}$ be a safe FO learning framework. For all $p \in \mathbb{N}^+$, $\mathfrak{F}$ is in* ELP(SbQ) *iff $\mathfrak{F}^p_\pi$ is in* ELP(SbQ) *and $\mathfrak{F}$ is in* ELP(SpQ) *iff $\mathfrak{F}^p_\pi$ is in* ELP(SpQ).

Proof. Let $\mathfrak{F} = (\mathcal{E}, \mathcal{L}, \mu)$, and $\mathfrak{F}_\pi = (\mathcal{E}_\pi, \mathcal{L}_\pi, \mu_\pi)$.

($\Rightarrow$) $\mathfrak{F}$ in ELP(SbQ) implies that there is a learning system $(L_\mathfrak{F}, T_\mathfrak{F})$ such that for any $\mathcal{K} \in \mathcal{L}$, $L_\mathfrak{F}$ asks only subset queries and $\mathcal{K}' \in (L_\mathfrak{F}, T_\mathfrak{F}(\mathcal{K}))(\Sigma_\mathcal{K})$ implies $\mathcal{K}' \equiv \mathcal{K}$. We consider the learning system $(L_{\mathfrak{F}_\pi}, T_{\mathfrak{F}_\pi}(\mathcal{T}))$ with input $\Sigma_\mathcal{T}$ and we explain the steps that the learner $L_{\mathfrak{F}_\pi}$ does to find the target. The learner will employ a binary search to find valuations in $\mathcal{T}^v$ in the interval $(0, 1]_p$. Claim A.4 states that for a given interval $[\beta, \alpha]_p$, it is possible to find the highest $\gamma \in [\beta, \alpha]_p$ such that $\gamma \in \mathcal{T}^v$ and a hypothesis $\mathcal{K}$ such that $\mathcal{K} \equiv \mathcal{T}^*_\gamma$.

Claim A.3. *Let $\mathcal{T} \in \mathcal{L}_\pi$ be the target. For any $\gamma \in [0, 1]_p$, the learner $L_{\mathfrak{F}_\pi}$ can find in*

polynomial time w.r.t. $|\mathcal{T}|$ *and the largest counterexample seen so far, a hypothesis* $\mathcal{K}$ *such that* $\mathcal{K} \equiv \mathcal{T}^*_\gamma$.

Proof. $L_{\mathfrak{F}_\pi}$ performs every step that the learner $L_{\mathfrak{F}}$ does in a path p in $\mathsf{T}_{L_{\mathfrak{F}},T_{\mathfrak{F}}(\mathcal{T}^*_\gamma),\Sigma_{\mathcal{T}^*}}$ (Remark 4.2). We can assume $\mathcal{T}^*_\gamma \in \mathcal{L}$ because $\mathfrak{F}$ is safe. Whenever in p there is a call to $\mathsf{SbQ}_{\mathfrak{F},\mathcal{T}^*_\gamma}$ with input $\mathcal{K}$, $L_{\mathfrak{F}_\pi}$ will call $\mathsf{SbQ}_{\mathfrak{F}_\pi,\mathcal{T}}$ with input $\mathcal{H} = \{(\phi,\gamma) \mid \phi \in \mathcal{K}\}$. By Point 3 of Proposition 4.7, $\mathcal{H} \models (\phi,\gamma)$ iff $\mathcal{H}^*_\gamma \models \phi$ and since $\mathcal{K} = \mathcal{H}^*_\gamma$, we get $\mathcal{H}^*_\gamma \models \phi$ iff $\mathcal{K} \models \phi$. Again by Point 3 of Proposition 4.7, $\mathcal{T} \models (\phi,\gamma)$ iff $\mathcal{T}^*_\gamma \models \phi$. By our consideration, $\mathcal{T} \not\models \mathcal{H}$ iff $\mathcal{H} \models (\phi,\gamma)$ and $\mathcal{T} \not\models (\phi,\gamma)$ iff $\mathcal{K} \models \phi$ and $\mathcal{T}^*_\gamma \not\models \phi$. Therefore, ϕ is a counterexample that can be returned by $\mathsf{SbQ}_{\mathfrak{F},\mathcal{T}^*_\gamma}$ with input $\mathcal{K}$ and $L_{\mathfrak{F}_\pi}$ continues the computation in p by considering ϕ as a counterexample. After a polynomial amount of time w.r.t. $|\mathcal{T}^*_\gamma|$ and the largest counterexample seen so far, $L_{\mathfrak{F}_\pi}$ will build $\mathcal{K}^\gamma$ such that $\mathcal{K}^\gamma \equiv \mathcal{T}^*_\gamma$. □

Claim A.4. *Let* $\beta, \alpha \in [0,1]_p$ *with* $\beta < \alpha$*, and let* $\mathcal{T} \in \mathcal{L}_\pi$ *be the target.* $L_{\mathfrak{F}_\pi}$ *can find in polynomial time w.r.t.* $|\mathcal{T}|$*,* p*, and the largest counterexample seen so far, the highest* $\gamma \in [\beta,\alpha]_p$ *such that* $\gamma \in \mathcal{T}^v \cup \{\beta\}$ *and a hypothesis* $\mathcal{K} \in \mathcal{L}$ *such that* $\mathcal{K} \equiv \mathcal{T}^*_\gamma$.

Proof. The idea is to search for the highest valuation in $\mathcal{T}^v$ with a binary-search like strategy. By Claim A.3, $L_{\mathfrak{F}_\pi}$ can find in polynomial time $\mathcal{K}^\alpha$ such that $\mathcal{K}^\alpha \equiv \mathcal{T}^*_\alpha$ and for $\gamma = \uparrow_p \left(\frac{\alpha+\beta}{2}\right)$ a $\mathcal{K}^\gamma$ such that $\mathcal{K}^\gamma \equiv \mathcal{T}^*_\gamma$.

If $\mathcal{K}^\alpha \not\models \mathcal{K}^\gamma$ (Remark 4.4), then there is a formula ϕ such that $\mathcal{T}^*_\alpha \not\models \phi$ and $\mathcal{T}^*_\gamma \models \phi$. By Point 3 of Proposition 4.7, $\mathcal{T} \models (\phi,\gamma)$ and by Points 4 and 5 of Proposition 4.7, $\gamma \leq \mathsf{val}(\phi,\mathcal{T}) \in \mathcal{T}^v$ and $\mathsf{val}(\phi,\mathcal{T}) < \alpha$. Therefore, there is a valuation $\delta = \mathsf{val}(\phi,\mathcal{T}) \in \mathcal{T}^v$ in the interval $[\gamma,\alpha)_p$. In this case, such valuation in $\mathcal{T}^v$ lies in the interval $[\gamma,\alpha)_p$. $L_{\mathfrak{F}_\pi}$ updates β to γ.

Otherwise, if $\mathcal{K}^\alpha \models \mathcal{K}^\gamma$ (Remark 4.4), no element in $\mathcal{T}^v$ is in $[\gamma,\alpha)_p$. For this reason, a valuation in $\mathcal{T}^v$ (if any) should be searched in the interval $[\beta,\gamma)_p$. $L_{\mathfrak{F}_\pi}$ updates α to γ.

After that, every step from the beginning is repeated with the updated α or β. When $\beta = \alpha = \gamma$, $L_{\mathfrak{F}_\pi}$ outputs the last computed $\mathcal{K}^\gamma$. This process requires $log_2[\beta,\alpha]_p \leq log_2[0,1]_p < 4p$ calls to Claim A.3, therefore $L_{\mathfrak{F}_\pi}$ terminates in polynomial time w.r.t. $|\mathcal{T}|$, p, and the largest counterexample seen so far. □

By Claim A.4, $L_{\mathfrak{F}_\pi}$ can find the highest $\alpha \in \mathcal{T}^v$ and $\mathcal{K}^\alpha$ such that $\mathcal{K}^\alpha \equiv \mathcal{T}^*_\alpha$. $L_{\mathfrak{F}_\pi}$ can then add $\mathcal{K}^\alpha$ to $\mathfrak{K}$ which is a set of classical hypotheses labelled with a valuation, initialised to the empty set. Again by Claim A.4, $L_{\mathfrak{F}_\pi}$ can find the highest $\beta \in \mathcal{T}^v$ and $\mathcal{K}^\beta$ such that $\mathcal{K}^\beta \equiv \mathcal{T}^*_\beta$ such that $\beta \in [0,\alpha)_p$ and add $\mathcal{K}^\beta$ to $\mathfrak{K}$. This process is repeated until

$\beta = 0$. In this last case, $L_{\mathfrak{F}_\pi}$ generates

$$\mathcal{H} = \bigcup_{\mathcal{K}^\gamma \in \mathfrak{K}} \{(\phi, \gamma) \mid \phi \in \mathcal{K}^\gamma\}$$

and it stops, writing $\mathcal{H}$ in the output tape. By Claim A.4, this process requires polynomial time w.r.t. p and $|\mathcal{T}^v|$ and the largest counterexample seen so far. Finally, by Lemma 4.9, we have that $\mathcal{H} \equiv \mathcal{T}$.

The proof when only superset queries can be asked, is analogous. The only difference is how responses to queries are used. Claim A.5 replaces Claim A.3 in the proof for superset queries.

Claim A.5. *Let $\mathcal{T} \in \mathcal{L}_\pi$ be the target. When only superset queries can be asked, for any $\gamma \in [0,1]_p$, we can find in polynomial time w.r.t. $|\mathcal{T}|$, p, and the largest counterexample seen so far a hypothesis $\mathcal{K}$ such that $\mathcal{K} \equiv \mathcal{T}^*_\gamma$.*

Proof. $\mathfrak{F}$ in ELP(SpQ) implies that there is a learning system $(L_\mathfrak{F}, T_\mathfrak{F})$ such that for any $\mathcal{K} \in \mathcal{L}$, $L_\mathfrak{F}$ asks only subset queries and $\mathcal{K}' \in (L_\mathfrak{F}, T_\mathfrak{F}(\mathcal{K}))(\Sigma_\mathcal{K})$ implies $\mathcal{K}' \equiv \mathcal{K}$. Let $\mathcal{T} \in \mathcal{L}_\pi$ be the target. $L_{\mathfrak{F}_\pi}$ performs every step that $L_\mathfrak{F}$ does in a path p in $\mathsf{T}_{L_\mathfrak{F}, T_\mathfrak{F}(\mathcal{T}^*_\gamma), \Sigma_{\mathcal{T}^*}}$ (Remark 4.2).

We can assume $\mathcal{T}^*_\gamma \in \mathcal{L}$ because $\mathfrak{F}$ is safe. Whenever in p there is a call to $\mathsf{SpQ}_{\mathfrak{F}, \mathcal{T}^*_\gamma}$ with input $\mathcal{K}$, $L_{\mathfrak{F}_\pi}$ will call $\mathsf{SpQ}_{\mathfrak{F}_\pi, \mathcal{T}}$ with input $\mathcal{H} = \{(\phi, \gamma) \mid \phi \in \mathcal{K}\}$. We know by Point 3 of Proposition 4.7 that $\mathcal{H} \models (\phi, \gamma)$ iff $\mathcal{H}^*_\gamma \models \phi$ iff (by construction) $\mathcal{K} \models \phi$. We also know again by Point 3 of Proposition 4.7, $\mathcal{T} \models (\phi, \gamma)$ iff $\mathcal{T}^*_\gamma \models \phi$. It follows that $\mathcal{H} \not\models \mathcal{T}$ iff $\mathcal{H} \not\models (\phi, \gamma)$ and $\mathcal{T} \models (\phi, \gamma)$ iff $\mathcal{K} \not\models \phi$ and $\mathcal{T}^*_\gamma \models \phi$. Therefore, ϕ is a counterexample that can be returned by $\mathsf{SpQ}_{\mathfrak{F}, \mathcal{T}^*_\gamma}$ with input $\mathcal{K}$, and $L_{\mathfrak{F}_\pi}$ continues the computation in p by considering ϕ as a counterexample. After a polynomial amount of time w.r.t. $|\mathcal{T}^*_\gamma|$ and the largest counterexample seen so far, $L_{\mathfrak{F}_\pi}$ will build $\mathcal{K}^\gamma$ such that $\mathcal{K}^\gamma \equiv \mathcal{T}^*_\gamma$. □

A similar claim to Claim A.4 can be stated where only superset queries can be asked and the rest of the proof follows the same strategy as in the case with only subset queries.

($\Leftarrow$) Let $\mathcal{K} \in \mathcal{L}$ be the target. Since $\mathfrak{F}_\pi$ is in ELP(SbQ), there is a learning system $(L_{\mathfrak{F}_\pi}, T_{\mathfrak{F}_\pi})$ such that for any $\mathcal{T} \in \mathcal{L}_\pi$, $L_{\mathfrak{F}_\pi}$ asks only subset queries, and $\mathcal{K}' \in (L_{\mathfrak{F}_\pi}, T_{\mathfrak{F}_\pi})(\Sigma_\mathcal{T})$ implies $\mathcal{K}' \equiv \mathcal{T}$. We consider the computation of the learning system $(L_\mathfrak{F}, T_\mathfrak{F}(\mathcal{K}))$ on input $\Sigma_\mathcal{K}$ where $T_\mathfrak{F}(\mathcal{K})$ is terminating.

By definition of $\mathfrak{F}_\pi$, we have that $\mathcal{T} = \{(\phi, 1) \mid \phi \in \mathcal{K}\} \in \mathcal{L}_\pi$. $L_\mathfrak{F}$ performs every step that $L_{\mathfrak{F}_\pi}$ does in a path p in $\mathsf{T}_{L_{\mathfrak{F}_\pi}, T_{\mathfrak{F}_\pi}(\mathcal{T}), \Sigma_\mathcal{T}}$ ($\Sigma_\mathcal{T}$ is built from the symbols of $\Sigma_\mathcal{K}$). Whenever in p there is a call to $\mathsf{SbQ}_{\mathfrak{F}_\pi, \mathcal{T}}$ with input $\mathcal{H}$, $L_\mathfrak{F}$ calls $\mathsf{SbQ}_{\mathfrak{F}, \mathcal{K}}$ with input $\mathcal{H}^*$.

Upon receiving a counterexample ϕ, $L_{\mathfrak{F}}$ knows $\mathcal{K} \not\models \phi$ and $\mathcal{H}^* \models \phi$. By construction we have $\mathcal{T}_1^* = \mathcal{K}$, and it follows that $\mathcal{T}_1^* \not\models \phi$ iff $\mathcal{K} \not\models \phi$. By Point 1 of Proposition 4.7, $\mathcal{T}_1^* \models \phi$ iff $\mathcal{T} \models (\phi, 1)$, therefore $L_{\mathfrak{F}}$ can continue the computation that $L_{\mathfrak{F}_\pi}$ performs with the counterexample $(\phi, 10^{-p}) \in \mathcal{E}_\pi$.

$L_{\mathfrak{F}_\pi}$ terminates in polynomial time w.r.t. $|\mathcal{T}|$ and the largest counterexample seen so far with output a hypothesis $\mathcal{H}$ that satisfies $\mathcal{H} \equiv \mathcal{T}$. As $\mathcal{H}^* \equiv \mathcal{T}^* = \mathcal{K}$, $L_{\mathfrak{F}}$ finds an equivalent hypothesis $\mathcal{H}^*$ w.r.t. the target $\mathcal{K}$. We have found a polynomial time learner for $\mathfrak{F}$ and we have that $\mathfrak{F}$ is in $\textsc{ELP}(\mathsf{SbQ})$ (translating counterexamples as described can be done in polynomial time w.r.t. the largest counterexample received so far).

The polynomial time reduction with only superset queries is analogous. Whenever $L_{\mathfrak{F}}$ should call $\mathsf{SpQ}_{\mathfrak{F}_\pi,\mathcal{T}}$ with input $\mathcal{H}$, the learner $L_{\mathfrak{F}}$ calls $\mathsf{SpQ}_{\mathfrak{F},\mathcal{K}}$ with input $\mathcal{H}^*$. Upon receiving a counterexample ϕ, $L_{\mathfrak{F}}$ knows $\mathcal{K} \models \phi$ and $\mathcal{H}^* \not\models \phi$. By construction we have $\mathcal{T}_1^* = \mathcal{K}$, and it follows that $\mathcal{T}_1^* \models \phi$ iff $\mathcal{K} \models \phi$. By Point 1 of Proposition 4.7, $\mathcal{T}_1^* \models \phi$ iff $\mathcal{T} \models (\phi, 1)$, therefore $L_{\mathfrak{F}}$ can continue the computation that $L_{\mathfrak{F}_\pi}$ performs with the counterexample $(\phi, 1) \in \mathcal{E}_\pi$. □

A.3 Proofs of Chapter 5

In this section, we provide the omitted proofs for the correctness of the HORN^* algorithm and the polynomial time transferability results between learning frameworks with membership and equivalence queries and frameworks with possibility and equivalence queries.

A.3.1 Horn

Lemma 5.4 (Frazier and Pitt [1993a] Lemma 4 Adaptation)**.** *Let $\mathcal{T}$ be the target Horn formula and $\mathcal{H}$ be the hypothesis built by HORN^*. At every step, it holds that $\mathcal{T} \models \mathcal{H}$.*

Proof. The algorithm adds a clause ϕ to the hypothesis after a membership query with input $\mathcal{I}$ returned 'no'. Such clause ϕ is such that $\mathsf{ant}(\phi)$ consists of literals set to true to the input interpretations $\mathcal{I}$ and $\mathsf{con}(\phi)$ are the only variables set to false by $\mathcal{I}$. As the membership query call returned a negative answer, we know that there is a clause $\mathcal{T} \models \psi$ such that $\mathsf{ant}(\psi) \subseteq \mathsf{ant}(\phi)$ and $\mathsf{con}(\psi) = \mathsf{con}(\phi)$. Therefore, $\mathcal{T} \models \phi$ for every $\phi \in \mathcal{H}$. □

Lemma 5.5 (Frazier and Pitt [1993a] Lemma 9 Adaptation)**.** *Let $\mathcal{T}$ be the target Horn formula. At any time during a run of* HORN^**, for any $\mathcal{A}_1, \mathcal{A}_2 \in S$, there are two distinct Horn rules $\mathsf{r}_1, \mathsf{r}_2 \in \mathcal{T}$ such that $\mathsf{ant}(\mathsf{r}_1) \subseteq \mathcal{A}_1$ and $\mathsf{ant}(\mathsf{r}_2) \subseteq \mathcal{A}_2$.*

Proof. The algorithm will add to the set of antecedents S only a set of variables $\mathcal{A}$ such that $\mathcal{T} \models \mathcal{A} \rightarrow \mathsf{v}$ for some $\mathsf{v} \in \mathsf{V}$. This follows by considering that in Line 8 only a set of variables appearing as an antecedent of a rule entailed by $\mathcal{T}$ is added by calling the membership oracle (with CON). Otherwise, by Lemma 5.4, we know that $\mathcal{I} \not\models \mathcal{T}$ and in Line 11 we add to S the set of all variables that can make the antecedent of a clause in $\mathcal{T}$ to 'true'.

We now show that two different elements $\mathcal{A}_1, \mathcal{A}_2 \in S$ are a superset of two different rule antecedents in $\mathcal{T}$. Line 11 is the only part where we add a new set $\mathcal{A}$ to the set S. This happens when we cannot find any other element in S that satisfies the condition in Line 8. That is, the last retuned counterexample $\mathcal{I}$ cannot be used to derive a set of variables mapped to 'true' $\mathcal{A}_t$ (Line 7) such that $\mathcal{A}_t$ intersected with any element in S identifies a smaller antecedent of a rule in $\mathcal{T}$ with respect to the antecedents already stored in S. Therefore, by Lemma 5.4, $\mathcal{A}_t \subseteq \mathsf{ant}(\mathsf{r})$ for a rule $\mathsf{r} \in \mathcal{T}$ such that no other $\mathcal{A} \in S$ satisfies $\mathcal{A} \subseteq \mathsf{ant}(\mathsf{r})$. On the other hand, when an element $\mathcal{A}' \in S$ is replaced with the set $\mathcal{A}$ after the check in Line 8, we either:

- can find at least two rules with different antecedents $\mathsf{r}_1, \mathsf{r}_2 \in \mathcal{T}$ such that $\mathcal{A} \subseteq \mathcal{A}' \subseteq \mathsf{ant}(\mathsf{r}_1)$, $\mathcal{A}' \subseteq \mathsf{ant}(\mathsf{r}_2)$, and $\mathsf{ant}(\mathsf{r}_1) \subset \mathsf{ant}(\mathsf{r}_2)$. Therefore, the algorithm will give precedence to the discovery of shorter antecedents first; or
- for a rule $\mathsf{r} \in \mathcal{T}$, $\mathcal{A} \subseteq \mathcal{A}' \subseteq \mathsf{ant}(\mathsf{r})$. So, this check allows to remove redundant variables in rules antecedents.

Therefore, the statement holds. □

Theorem 5.6 (Frazier and Pitt [1993a] Theorem 10 Adaptation)**.** *Let V be a finite set of propositional variables and let $\mathcal{T}$ be the unknown target Horn formula.* HORN^* *runs in polynomial time with respect to $|\mathcal{T}|$, and $|\mathsf{V}|$, and outputs a hypothesis $\mathcal{H}$ such that $\mathcal{H} \equiv \mathcal{T}$ by asking membership and equivalence queries.*

Proof. By Lemma 5.5, the size of S is bounded by the number of rules in $\mathcal{T}$. Moreover, the algorithm only replaces elements in S with proper subsets of themselves. As this can happen at most $|\mathsf{V}|$ times for each element in S, the main loop terminates after a polynomial number of iterations with respect to $|\mathcal{T}|$ and $|\mathsf{V}|$. Finally, the algorithm will output $\mathcal{H} \equiv \mathcal{T}$ by definition of equivalence query oracle. □

A.3.2 Results with Possibility Queries

In this section, we show the theoretical results that motivated the development of the Π_HORN* algorithm. They hold in more general settings when examples are partial interpretations.

With Theorem A.6 we show one direction of Theorem 5.13.

Theorem A.6. *Let $\mathfrak{F}$ be a learning framework. If $\mathfrak{F}_\pi$ is polynomial time learnable with possibility and equivalence queries, then $\mathfrak{F}$ is polynomial time learnable with membership and equivalence queries.*

Proof. Let $\mathfrak{F} = (\mathcal{E}, \mathcal{H}, \mu)$ and let $\mathcal{K} \in \mathcal{H}$ be the target. Since $\mathfrak{F}_\pi$ is polynomial time learnable, the restriction of $\mathfrak{F}_\pi$ for the case in which all valuations are 1, denoted $\mathfrak{F}^1_\pi$, is also polynomial time learnable. Then, there is a polynomial time learner A_π for $\mathfrak{F}^1_\pi = (\mathcal{E}_\pi, \mathcal{H}_\pi, \mu_\pi)$. We start the execution of A_π that attempts to learn a hypothesis $\mathcal{H}$ equivalent to $\mathcal{T} = \{(\phi, 1) \mid \phi \in \mathcal{K}\} \in \mathcal{H}_\pi$.

For all $(\mathcal{I}, \alpha) \in \mathcal{E}_\pi$, a call to $\mathsf{PQ}_{\mathfrak{F}_\pi, \mathcal{T}}$ with input $\mathcal{I}$ returns 1 iff $\mathcal{I} \models \mathcal{T}_1^*$ (recall that $\mathcal{T}^* = \mathcal{K}$) iff $\mathcal{I} \models \mathcal{K}$ iff $\mathsf{MQ}_{\mathfrak{F},\mathcal{K}}(\mathcal{I})$ returns 'yes'. Also, we can simulate a call to $\mathsf{EQ}_{\mathfrak{F}_\pi, \mathcal{T}}$ with $\mathcal{H}$ as input by calling $\mathsf{EQ}_{\mathfrak{F},\mathcal{K}}$ with $\mathcal{H}^*$ as input. By construction for all $\mathcal{I} \in \mathcal{E}$, we get $\mathcal{I} \models \mathcal{K}$ iff $\mathcal{I} \models \mathcal{T}_1^*$ iff $\pi_{\mathcal{T}}(\mathcal{I}) = 1$.

Whenever A_π asks a possibility query with input $(\mathcal{I}, \alpha)$, we call $\mathsf{MQ}_{\mathfrak{F},\mathcal{K}}$ with input $\mathcal{I}$. If the answer is 'yes', we return 1 to A_π, otherwise 0. Whenever A_π asks an equivalence query with input $\mathcal{H}$, we call $\mathsf{EQ}_{\mathfrak{F},\mathcal{K}}$ with input $\mathcal{H}^*$. Upon receiving a negative counterexample $\mathcal{I}$ for $\mathcal{K}$, we know that $\mathcal{I} \not\models \mathcal{K}$, and $\mathcal{I} \models \mathcal{H}^*$. This means that $\pi_{\mathcal{H}}(\mathcal{I}) = 1$. By construction, $\mathcal{K} = \mathcal{T}^*$, $\mathcal{I} \not\models \mathcal{T}_1^*$, and so $\pi_{\mathcal{T}}(\mathcal{I}) = 0$. Therefore we return the negative counterexample $(\mathcal{I}, 0)$ to A_π.

Upon receiving a positive counterexample $\mathcal{I}$ for $\mathcal{K}$, we know that $\mathcal{I} \models \mathcal{K}$, and $\mathcal{I} \not\models \mathcal{H}^*$. This means that $\pi_{\mathcal{H}}(\mathcal{I}) < 1$. Since by construction $\mathcal{K} = \mathcal{T}^*$, $\mathcal{I} \models \mathcal{T}_1^*$, and so $\pi_{\mathcal{T}}(\mathcal{I}) = 1$. We return the positive counterexample $(\mathcal{I}, 1)$ to A_π.

Eventually, A_π will output a hypothesis $\mathcal{H} \equiv \mathcal{T}$ in polynomial time w.r.t. $|\mathcal{T}|$ and the largest counterexample received so far. Clearly, $\mathcal{H}^*$ is as required. □

The converse of Theorem A.6 does not hold. The argument is similar to the one shown in Theorem 4.21. That is, simple strategies to find a hypothesis may not work when the learning framework is extended with possibilistic valuations because the algorithms also have to deal with multiple valuations (Example 4.20).

We now show the other direction of Theorem 5.13.

Theorem A.7. *Let $\mathfrak{F}$ be a safe learning framework. If $\mathfrak{F}$ is polynomial time learnable with membership and equivalence queries, then $\mathfrak{F}_\pi$ is polynomial time learnable with possibility and equivalence queries.*

Proof. Let $\mathfrak{F}_\pi = (\mathcal{E}_\pi, \mathcal{H}_\pi)$ and let $\mathcal{T} \in \mathcal{H}_\pi$ be the target. We are going to use the following lemma, presented in Chapter 4.

Lemma 4.9. *Let $\mathcal{T}$ be a possibilistic KB. Let I be a finite set of valuations such that $\mathcal{T}^v \subseteq I$. If for each $\alpha \in I$ there is some FO KB $\mathcal{K}^*_\alpha$ such that $\mathcal{K}^*_\alpha \equiv \mathcal{T}^*_\alpha$ then, it holds that $\mathcal{T} \equiv \{(\phi, \alpha) \mid \phi \in \mathcal{K}^*_\alpha, \alpha \in I\}$.*

Strategy. Let A be a polynomial time learner[2] for $\mathfrak{F}$. We run multiple instances of A and we denote by R the set of run instances of A. Each instance in R is denoted A_β and attempts to learn a hypothesis equivalent to $\mathcal{T}_\beta$, with $\beta \in (0, 1]$. We denote by $\mathcal{K}^{\beta,n}$ the hypothesis given as input by A_β when it asks its n-th equivalence query. For $n = 0$, we assume that $\mathcal{K}^{\beta,n} = \emptyset$. We omit the number n from $\mathcal{K}^{\beta,n}$ if we want to refer to the hypothesis built by A_β when asking its last equivalence query. Initially, $\mathsf{R} := \{A_1\}$.

Simulating a membership query. Whenever $A_\beta \in \mathsf{R}$ asks a query with input $\mathcal{I} \in \mathcal{E}$, we can simulate $\mathsf{MQ}_{\mathfrak{F},\mathcal{T}^*_\beta}$ by calling $\mathsf{PQ}_{\mathfrak{F}_\pi,\mathcal{T}}$ with input $\mathcal{I}$. If the answer $\pi_\mathcal{T}(\mathcal{I}) = \delta$ is greater than $1 - \beta$, by Point 2 of Proposition 4.7, we return 'yes' to A_β, otherwise we return 'no'.

Simulating an equivalence query. Whenever $A_\beta \in \mathsf{R}$ asks its n-th equivalence query, we leave A_β waiting in the query state. When all $A_\alpha \in \mathsf{R}$ are waiting in the query state, we create

$$\mathcal{H} := \bigcup_{A_\alpha \in \mathsf{R}} \{(\phi, \alpha) \mid \phi \in \mathcal{K}^\alpha\} \tag{A.1}$$

and call $\mathsf{EQ}_{\mathfrak{F}_\pi,\mathcal{T}}$ with $\mathcal{H}$ as input (note: each instance $A_\alpha \in \mathsf{R}$ may have asked a different number of equivalence queries when another A_β asks its n-th equivalence query). If the answer is 'yes', we have computed $\mathcal{H}$ such that $\mathcal{H} \equiv \mathcal{T}$ and we are done.

Upon receiving a positive counterexample $(\mathcal{I}, \gamma)$, by definition, we know that $\mathcal{I} \models \mathcal{T}^*_{1-\eta}$ for all $\gamma \geq \eta \in [0, 1]$. Since $(\mathcal{I}, \gamma)$ is a positive counterexample and by definition of $\pi_\mathcal{H}$, we also know that $\pi_\mathcal{H}(\mathcal{I}) \geq \gamma$ and $\mathcal{I} \not\models \mathcal{H}^*_{1-\gamma}$. This means that for some $(\phi, 1 - \delta) \in \mathcal{H}$ with $(1-\delta) \geq (1-\gamma)$, $\mathcal{I} \not\models \phi$ and hence $\mathcal{I} \not\models \mathcal{H}^*_{1-\delta}$. By construction of $\mathcal{H}$, each valuation

[2]Assume w.l.o.g. that A always eventually ask an equivalence query until it finds an equivalent hypothesis (but may execute other steps and ask membership queries between each equivalence query).

$\alpha \in \mathcal{H}^v$ is associated a $A_\alpha \in \mathsf{R}$. As a consequence, we send to such $A_{1-\delta} \in \mathsf{R}$ the positive counterexample $\mathcal{I}$ and $A_{1-\delta}$ resumes its execution.

If $(\mathcal{I}, \gamma)$ is a negative counterexample, we know that $\mathcal{I} \not\models \mathcal{T}^*_{1-\gamma}$. We call $\mathsf{PQ}_{\mathfrak{F}_\pi, \mathcal{T}}$ to obtain $\pi_{\mathcal{T}}(\mathcal{I}) = \delta$. If $A_{1-\delta} \notin \mathsf{R}$, we start the execution of the instance $A_{1-\delta}$ of algorithm A and add $A_{1-\delta}$ to R. Otherwise, we send the negative counterexample $\mathcal{I}$ to $A_{1-\delta}$ that resumes its execution.

Termination. We now argue that this procedure terminates in polynomial time w.r.t. $|\mathcal{T}|$, and the largest counterexample seen so far. Since there is only one instance A_β in R for each valuation $\beta \in \mathcal{T}^v$ such that there is at least a formula $\phi \in \mathcal{E}$ satisfying $\beta = \mathsf{val}(\phi, \mathcal{T})$, by Point 1 of Proposition 4.7, we have that at all times $|\mathsf{R}|$ is linear in $|\mathcal{T}^v|$, which is bounded by $|\mathcal{T}|$. Since $\mathfrak{F}$ is safe and A is a polynomial time learner for $\mathfrak{F}$, each $A_\beta \in \mathsf{R}$ terminates, in polynomial time in the size of $\mathcal{T}^*_\beta$ and the largest counterexample seen so far, and outputs $\mathcal{K}^\beta \equiv \mathcal{T}_\beta$. By Claim 4.9 and by construction of $\mathcal{H}$, $\mathcal{H} \equiv \mathcal{T}$ and the process terminates. As a consequence, $\mathfrak{F}_\pi$ is polynomial time learnable. □

Theorems A.6 and A.7 directly imply Theorem 5.13.

Theorem 5.13. *Let $\mathfrak{F}$ be a safe learning framework. $\mathfrak{F}_\pi$ is exactly learnable in polynomial time with possibility and equivalence queries iff $\mathfrak{F}$ is exactly learnable in polynomial time with membership and equivalence queries.*

A.4 Possibility and Necessity as Upper and Lower Probabilities

Possibilistic logic can be also used to reason about imprecise probabilities [Coolen et al.]. We introduced this relationship in Section 2.2 in Chapter 2 when we stated the principle 'what is probable must be possible and something necessarily true must be probable'. A possibility distribution π defines the pair of measures (N_π, Π_π) that characterise a class P of probability distributions over $\mathcal{E}$ that satisfy the constraints

$$\mathsf{P} = \{\mathcal{D} \mid \phi \in \mathcal{E}, N(\phi) \leq \mathcal{D}(\phi) \leq \Pi(\phi)\}.$$

The modelling of uncertain probabilities with possibilistic logic is straightforward with *normalised* possibility distributions. That is, distributions that assign to at least an element $\mathcal{I} \in \Omega$ the maximal possibility value, $\pi(\mathcal{I}) = 1$. This is because a normalised π has a nice property that ensures that the possibility value is always higher or equal to the

v_1	v_2	$(\mathsf{v}_1, 0.2)$	$(\neg\mathsf{v}_1, 0.8)$	$(\mathsf{v}_2, 0.9)$	$(\neg\mathsf{v}_2, 0.1)$	π
1	1	1	0.2	1	0.9	0.2
1	0	1	0.2	0.1	1	0.1
0	1	0.8	1	1	0.9	0.8
0	0	0.8	1	0.1	1	0.1

Figure A.1: Possibilistic truth table of $\mathcal{H}$.

necessity value of a formula. Indeed, for every $\phi \in \mathcal{E}$, it holds that $\Pi(\phi) = 1$ or $N(\phi) = 0$ and it trivially holds that $N(\phi) \leq \Pi(\phi)$. With normalised possibility distributions, we can identify two cases for a formula $\phi \in \mathcal{E}$:

1. if $\Pi(\phi) = 1$, then $N(\neg\phi) = 0$. We are uncertain about the values of $N(\phi)$ and $\Pi(\neg\phi)$ and with the necessity valued constraint w.r.t. ϕ $(N(\phi) \geq \alpha)$, we can bind the lower bound of the probability associated to ϕ and the upper bound of the probability associated to $\neg\phi$. The more $N(\phi)$ is close to 1, the smaller the value of $\Pi(\neg\phi)$ becomes;

2. if $\Pi(\neg\phi) = 1$, then $N(\phi) = 0$ and we are in a similar case as before. The constraint $N(\neg\phi)$ will restrict the interval that defines the family of probability measures induced by the possibility distribution.

If the normalisation assumption is removed, not every possibility distribution can be used to model uncertain probabilities as just described. Example A.8 shows a case where this relationship cannot hold.

Example A.8. We consider the propositional language with only two variables p, q and let π be the possibility distribution induced by the possibilistic KB

$$\mathcal{H} = \{(\mathsf{v}_1, 0.2), (\neg\mathsf{v}_1, 0.8), (\mathsf{v}_2, 0.9), (\neg\mathsf{v}_2, 0.1)\}.$$

As shown in the Figure A.1, we have that $\Pi(\mathsf{v}_2) = 0.8$ but $N(\mathsf{v}_2) = 0.9$, also $\Pi(\neg\mathsf{v}_2) = 0.1$ but $N(\neg\mathsf{v}_2) = 0.2$. As $\Pi(\mathsf{v}_2) < N(\mathsf{v}_2)$, this possibility distribution cannot be used to model uncertain probabilities as described. This case may happen when the possibility distribution is not normalised ($\mathcal{H}$ is partially inconsistent). ◁

When the inconsistency level of the possibilistic KB $\mathcal{H}$ is different from 0, the least specific possibility distribution $\pi_{\mathcal{H}}$ is not normalised. Therefore, the possibility distribution cannot be directly used to define a family of probability distributions lying between N and Π. If there is no formula ϕ with $\Pi(\phi) = 1$, we cannot guarantee the property that $N(\phi) \leq \Pi(\phi)$ (Example A.8). A solution to this problem, would be to subtract to the value of the necessity measure of a formula the value representing the inconsistency level

$\alpha = \mathsf{inc}(\mathcal{H})$ of the KB $\mathcal{H}$ into consideration to ensure the property $N(\phi) - \alpha \leq \Pi(\phi)$. This idea works as proved, with the help of Proposition A.9, in Proposition A.10.

Proposition A.9. *If the KB $\mathcal{H}$ satisfies $\mathsf{inc}(\mathcal{H}) = 0$, then $\pi_{\mathcal{H}}$ is normalised.*

Proof. If $\mathsf{inc}(\mathcal{H}) = 0$, there is at least a model $\mathcal{I}$ for $\mathcal{H}^*$. By definition of the least specific possibility distribution, $\pi_{\mathcal{H}}(\mathcal{I}) = 1$ since $\mathcal{I}$ satisfies every formula in $\mathcal{H}^*$. □

Recall that as we defined in Section 2.3 in Chapter 2, with the KB $\mathcal{H}$ we write $\Pi_{\mathcal{H}}$ and $N_{\mathcal{H}}$ to denote the possibility and necessity measure associated to the possibilistic distribution $\pi_{\mathcal{H}}$.

Proposition A.10. *Let $\mathcal{H}$ be a possibilistic KB with $\alpha = \mathsf{inc}(\mathcal{H})$. For any formula ϕ expressible in the language of $\mathcal{H}$, $N_{\mathcal{H}}(\phi) - \alpha \leq \Pi_{\mathcal{H}}(\phi)$.*

Proof. If $\alpha = 0$, by Proposition A.9 π is normalised, and it holds that $\Pi_{\mathcal{H}}(\phi) = 1$ or $N_{\mathcal{H}}(\phi) = 0$ and for every formula ϕ, $N_{\mathcal{H}}(\phi) \leq \Pi_{\mathcal{H}}(\neg\phi)$. To continue we use the following observation.

Claim A.11. *Let $\alpha = \mathsf{inc}(\mathcal{H})$. If for a formula ϕ, $\Pi_{\mathcal{H}_{\overline{\alpha}}}(\phi) = 1$, then $\Pi_{\mathcal{H}}(\phi) = 1 - \alpha$.*

Proof. For any $\beta > \alpha$ there is an interpretation $\mathcal{I}$ in the set Ω of interpretations in the considered logic language, such that $\mathcal{I} \models \mathcal{H}^*_\beta$ and $\mathcal{I} \not\models \mathcal{H}^*_\alpha$ (because $\mathcal{H}^*_{\overline{\alpha}}$ is consistent). We also know that α is the highest valuation of a formula $(\psi, \alpha) \in \mathcal{H}^*_\alpha$ such that $\mathcal{I} \not\models \psi$, therefore $\pi_{\mathcal{H}}(\mathcal{I}) = 1 - \alpha$. Since $\alpha = \mathsf{inc}(\mathcal{H})$, then $\pi_{\mathcal{H}}$ associates to any interpretation a value less or equal $1 - \alpha$. That is, there is no other interpretation $\mathcal{I}'$ with $\pi_{\mathcal{H}}(\mathcal{I}') > \pi_{\mathcal{H}}(\mathcal{I}) = 1 - \alpha$. By definition, it means that $\Pi_{\mathcal{H}}(\phi) = \sup_{\mathcal{I}\in\Omega}\{\pi_{\mathcal{H}}(\mathcal{I})\} = 1 - \alpha$. □

By Proposition A.9, $\pi_{\mathcal{H}_{\overline{\alpha}}}$ is normalised and we can distinguish two cases:

1. $\Pi_{\mathcal{H}_{\overline{\alpha}}}(\phi) = 1$ and $N_{\mathcal{H}_{\overline{\alpha}}}(\neg\phi) = 0$;
2. $\Pi_{\mathcal{H}_{\overline{\alpha}}}(\neg\phi) = 1$ and $N_{\mathcal{H}_{\overline{\alpha}}}(\phi) = 0$.

Case 1. Assume $\Pi_{\mathcal{H}_{\overline{\alpha}}}(\phi) = 1$ and then $N_{\mathcal{H}_{\overline{\alpha}}}(\neg\phi) = 0$. As $\alpha = \mathsf{inc}(\mathcal{H})$, we have that

$$\Pi_{\mathcal{H}_{\overline{\alpha}}}(\phi) = \sup_{\mathcal{I}\in\Omega}\{\pi_{\mathcal{H}_{\overline{\alpha}}}(\mathcal{I}) \mid \mathcal{I} \models \phi\} = 1 \overset{\text{Claim A.11}}{=} \Pi_{\mathcal{H}}(\phi) + \alpha$$

and as a consequence, $N_{\mathcal{H}}(\neg\phi) = \alpha$.

If $N_{\mathcal{H}_{\overline{\alpha}}}(\phi) = \beta$, with $\beta > \alpha \geq 0$, we know that $\Pi_{\mathcal{H}_{\overline{\alpha}}}(\neg\phi) < 1$. It follows that every interpretation that satisfies $\neg\phi$ falsifies at least one formula ψ such that $(\psi, \delta) \in \mathcal{H}_{\overline{\alpha}}$. Then, it holds that

$$\Pi_{\mathcal{H}_{\overline{\alpha}}}(\neg\phi) = \sup_{\mathcal{I}\in\Omega}\{1 - \sup\{\delta \mid (\psi, \delta) \in \mathcal{H} \text{ and } \mathcal{I} \models \neg\psi \text{ and } \delta > \alpha\} \mid \mathcal{I} \models \neg\phi\} = \Pi_{\mathcal{H}}(\neg\phi).$$

Therefore, it follows that $N_{\mathcal{H}}(\phi) = \beta$. With our observations, we see that

$$N_{\mathcal{H}}(\phi) = N_{\mathcal{H}_{\overline{\alpha}}}(\phi) \leq 1 = \Pi_{\mathcal{H}_{\overline{\alpha}}}(\phi) = \Pi_{\mathcal{H}}(\phi) + \alpha,$$

$$N_{\mathcal{H}}(\neg\phi) - \alpha = 0 \leq \Pi_{\mathcal{H}}(\neg\phi) = 1 - \beta.$$

Case 2. If $N_{\mathcal{H}_{\overline{\alpha}}}(\phi) = \beta$, with $\beta \leq \alpha$, then $\Pi_{\mathcal{H}_{\overline{\alpha}}}(\neg\phi) \geq 1 - \alpha$. This means that every interpretation that satisfies $\neg\phi$ falsifies (if any) a formula $(\psi, \delta) \in \mathcal{H}_{\overline{\alpha}}$ with $\delta \leq \alpha$.

$$\Pi_{\mathcal{H}_{\overline{\alpha}}}(\neg\phi) = \sup_{\mathcal{I}\in\Omega}\{1 - \sup\{\delta \mid (\psi, \delta) \in \mathcal{H} \text{ and } \mathcal{I} \models \neg\psi \text{ and } \delta \leq \alpha\} \mid \mathcal{I} \models \neg\phi\} = 1.$$

Therefore, $\beta = 0$ because there is no $(\psi, \delta) \in \mathcal{H}_{\overline{\alpha}}$ with $\delta \leq \alpha$. It follows that $\Pi_{\mathcal{H}}(\neg\phi) = 1 - \alpha$ and $N_{\mathcal{H}}(\phi) = \alpha$. Also in this case, $0 = N_{\mathcal{H}}(\phi) - \alpha \leq \Pi_{\mathcal{H}}(\phi)$, and since we shown earlier that $N_{\mathcal{H}}(\neg\phi) = \alpha$, then $N_{\mathcal{H}}(\neg\phi) - \alpha = 0 \leq \Pi_{\mathcal{H}}(\neg\phi)$.

The case $\Pi_{\mathcal{H}_{\overline{\alpha}}}(\neg\phi) = 1$ and $N_{\mathcal{H}_{\overline{\alpha}}}(\phi) = 0$ is similar to the previous point and in both cases the statement holds. □

Recall the relationship between the possibility measure and uncertain probabilities. To know the lower bound of the probability of a proposition ϕ to hold, we need to check $N(\phi) = \mathsf{val}(\phi, \mathcal{H})$ and subtract $\mathsf{inc}(\mathcal{H})$. The upper bound of the probability associated to ϕ is instead $\Pi(\phi) = 1 - N(\neg\phi) = 1 - \mathsf{val}(\neg\phi, \mathcal{H})$ (Example A.12).

Example A.12. Let $\mathcal{H} = \{(\mathsf{v}_1 \vee \mathsf{v}_2, 0.8), (\neg\mathsf{v}_1 \vee \neg\mathsf{v}_2, 0.3)\}$. The least specific possibility distribution associated to $\mathcal{H}$ defines the family P of probability distributions on interpretations with variables $\mathsf{v}_1, \mathsf{v}_2$ such that for every $\mathcal{D} \in \mathsf{P}$, we obtain

$$\begin{aligned} 0 = N(\mathsf{v}_1) \leq &\mathcal{D}(\mathsf{v}_1) \leq \Pi(\mathsf{v}_1) = 1 \\ 0 = N(\neg\mathsf{v}_1 \wedge \neg\mathsf{v}_2) \leq &\mathcal{D}(\neg\mathsf{v}_1 \wedge \neg\mathsf{v}_2) \leq \Pi(\neg\mathsf{v}_1 \wedge \neg\mathsf{v}_2) = 0.2 \\ 0.8 = N(\mathsf{v}_1 \vee \mathsf{v}_2) \leq &\mathcal{D}(\mathsf{v}_1 \vee \mathsf{v}_2) \leq \Pi(\mathsf{v}_1 \vee \mathsf{v}_2) = 1 \end{aligned}$$

that are the related probabilities of a formula to hold, and so on. ◁

With our considerations, once we have a set of possibilistic formulas we may be able to carry reasoning over uncertain probabilities by querying the KB $\mathcal{H}$ what is the necessity and possibility value of a fact (minimum and maximum probability of a fact).

Bibliography

R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. *Sigmod Record*, 22, 1993. doi: 10.1145/170036.170072.

V. Alekseev. On the number of intersection semilattices [in Russian]. *DiskretnayaMat.1*, page 129–136, 1989.

D. Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, Apr 1988. doi: 10.1023/A:1022821128753. URL `https://doi.org/10.1023/A:1022821128753`.

D. Angluin, M. Frazier, and L. Pitt. Learning conjunctions of Horn clauses. *Machine Learning*, 9:147–164, 1992.

G. Antipov, M. Baccouche, and J.-L. Dugelay. Face aging with conditional generative adversarial networks. *2017 IEEE International Conference on Image Processing (ICIP)*, pages 2089–2093, 2017.

M. Arias and J. L. Balcázar. Canonical Horn representations and query learning. In *ALT*, pages 156–170. Springer, 2009.

M. Arias and R. Khardon. Learning closed Horn expressions. *Inf. Comput.*, 178(1): 214–240, 2002.

M. Arias, R. Khardon, and J. Maloberti. Learning Horn expressions with LOGAN-H. *J. Mach. Learn. Res.*, 8:549–587, 2007.

H. Arimura. Learning acyclic first-order Horn sentences from entailment. In *ALT*, volume 1316 of *Lecture Notes in Computer Science*, pages 432–445. Springer, 1997a.

H. Arimura. Learning acyclic first-order Horn sentences from entailment. In *International Workshop on Algorithmic Learning Theory*, pages 432–445, 1997b.

D. Baskakov and D. Arseniev. On the computational complexity of deep learning algorithms. In *Proceedings of International Scientific Conference on Telecommunications, Computing and Control*, pages 343–356, Singapore, 2021. Springer Singapore. ISBN 978-981-33-6632-9. doi: https://doi.org/10.1007/978-981-33-6632-9_30.

J. L. J. L. Bell and M. Machover. *A course in mathematical logic / by J. L. Bell and M. Machover.* North-Holland, Amsterdam, 1977. ISBN 0720428440.

S. Benferhat, D. Dubois, and H. Prade. Representing default rules in possibilistic logic. In *International Conference on Principles of Knowledge Representation and Reasoning*, 1992.

A. L. Blum. Separating distribution-free and mistake-bound learning models over the boolean domain. *SIAM J. Comput.*, 23(5), 1994.

L. Bottou. Stochastic learning. In *Advanced Lectures on Machine Learning*, Lecture Notes in Artificial Intelligence, LNAI 3176, pages 146–168. Springer Verlag, Berlin, 2004. doi: 10.1007/978-3-540-28650-9_7.

I. Bratko. Applications of machine learning: Towards knowledge synthesis. *New Generation Computing*, 11(3):343–360, Sep 1993. ISSN 1882-7055. doi: 10.1007/BF03037182. URL `https://doi.org/10.1007/BF03037182`.

L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and Regression Trees.* Taylor & Francis, Andover, England, UK, 1984. ISBN 978-0-41204841-8.

N. H. Bshouty and C. A. Haddad-Zaknoon. Adaptive exact learning of decision trees from membership queries. In *ALT*, volume 98 of *Proceedings of Machine Learning Research*, pages 207–234. PMLR, 2019.

G. Burosch, J. Demetrovics, G. Katona, D. Kleitman, and A. Sapozhenko. On the number of closure operations. pages 91–105, Budapest, 1993. János Bolyai Mathematical Society.

M. Campbell, A. J. H. Jr., and F. Hsu. Deep blue. *Artif. Intell.*, 134(1-2):57–83, 2002. doi: 10.1016/S0004-3702(01)00129-1.

F. Chollet et al. Keras. GitHub, 2015. URL `https://github.com/fchollet/keras`.

P. Cimiano, J. Völker, and P. Buitelaar. Ontology construction. In *Handbook of Natural Language Processing, Second Edition.*, pages 577–604. Chapman and Hall/CRC, 2010.

W. W. Cohen. Fast effective rule induction. In *Machine Learning, Proceedings of the Twelfth International Conference on Machine Learning, Tahoe City, California, USA, July 9-12, 1995*, pages 115–123. Morgan Kaufmann, 1995. doi: 10.1016/b978-1-55860-377-6.50023-2.

W. W. Cohen and H. Hirsh. The learnability of description logics with equality constraints. *Mach. Learn.*, 17(2-3):169–199, 1994.

F. P. A. Coolen, M. C. M. Troffaes, and T. Augustin. *Imprecise Probability*, pages 645–648. Berlin, Heidelberg. ISBN 978-3-642-04898-2. doi: 10.1007/978-3-642-04898-2_296. URL https://doi.org/10.1007/978-3-642-04898-2_296.

S. Dash, O. Günlük, and D. Wei. Boolean decision rules via column generation. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 4660–4670, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/743394beff4b1282ba735e5e3723ed74-Abstract.html.

L. De Raedt. Logical settings for concept-learning. *Artificial Intelligence*, 95(1):187–201, 1997. ISSN 0004-3702. doi: 10.1016/S0004-3702(97)00041-6.

J. A. Dhanraj, M. Prabhakar, C. P. Ramaian, M. Subramaniam, J. M. Solomon, and N. Vinayagam. Increasing the Wind Energy Production by Identifying the State of Wind Turbine Blade. In *Lecture Notes in Mechanical Engineering*, pages 139–148. Springer Nature Singapore, 2022. doi: 10.1007/978-981-16-7909-4_13.

Y. Djouadi, D. Dubois, and H. Prade. Possibility theory and formal concept analysis: Context decomposition and uncertainty handling. pages 260–269, 06 2010. doi: 10.1007/978-3-642-14049-5_27.

C. Domingo, N. Mishra, and L. Pitt. Efficient read-restricted monotone CNF/DNF dualization by learning with membership queries. *Mach. Learn.*, 37(1):89–110, 1999.

P. Domingos and G. Hulten. Mining high-speed data streams. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '00, page 71–80, New York, NY, USA, 2000a. Association for Computing Machinery. ISBN 1581132336. doi: 10.1145/347090.347107.

P. M. Domingos and G. Hulten. Mining high-speed data streams. In *KDD*, pages 71–80. ACM, 2000b. doi: 10.1145/347090.347107.

W. F. Dowling and J. H. Gallier. Linear-time algorithms for testing the satisfiability of propositional Horn formulae. *The Journal of Logic Programming*, 1(3):267–284, 1984. ISSN 0743-1066. doi: https://doi.org/10.1016/0743-1066(84)90014-1. URL https://www.sciencedirect.com/science/article/pii/0743106684900141.

P. N. Druzhkov and V. D. Kustikova. A survey of deep learning methods and software tools for image classification and object detection. *Pattern Recognition and Image Analysis*, 26(1):9–15, Jan 2016. ISSN 1555-6212. doi: 10.1134/S1054661816010065. URL https://doi.org/10.1134/S1054661816010065.

D. Dua and C. Graff. UCI machine learning repository, 2017.

M. R. C. Duarte, B. Konev, and A. Ozaki. ExactLearner: A tool for exact learning of EL ontologies. In *KR*, pages 409–414. AAAI Press, 2018.

D. Dubois and H. Prade. The principle of minimum specificity as a basis for evidential reasoning. In *Uncertainty in Knowledge-Based Systems*, pages 75–84, Berlin, Heidelberg, 1987. Springer Berlin Heidelberg. ISBN 978-3-540-48020-4.

D. Dubois and H. Prade. Modelling uncertainty and inductive inference: A survey of recent non-additive probability systems. *Acta Psychologica*, 68(1):53–78, 1988. ISSN 0001-6918. doi: https://doi.org/10.1016/0001-6918(88)90045-5. URL `https://www.sciencedirect.com/science/article/pii/0001691888900455`.

D. Dubois and H. Prade. Resolution principles in possibilistic logic. *Int. J. Approx. Reasoning*, 4(1):1–21, 1990a.

D. Dubois and H. Prade. Consonant approximations of belief functions. *International Journal of Approximate Reasoning*, 4(5):419–449, 1990b. ISSN 0888-613X. doi: https://doi.org/10.1016/0888-613X(90)90015-T. URL `https://www.sciencedirect.com/science/article/pii/0888613X9090015T`.

D. Dubois and H. Prade. When upper probabilities are possibility measures. *Fuzzy Sets and Systems*, 49:65–74, 1992. doi: https://doi.org/10.1016/0165-0114(92)90110-P.

D. Dubois and H. Prade. *Possibility Theory: Qualitative and Quantitative Aspects*, pages 169–226. Springer Netherlands, Dordrecht, 1998. ISBN 978-94-017-1735-9. doi: 10.1007/978-94-017-1735-9_6. URL `https://doi.org/10.1007/978-94-017-1735-9_6`.

D. Dubois and H. Prade. Possibility theory, probability theory and multiple-valued logics: A clarification. *Ann. Math. Artif. Intell.*, 32(1-4):35–66, 2001.

D. Dubois and H. Prade. Possibilistic logic - an overview. In *Computational Logic*, pages 283–342. 2014. doi: 10.1016/B978-0-444-51624-4.50007-1. URL `https://doi.org/10.1016/B978-0-444-51624-4.50007-1`.

D. Dubois and H. Prade. Possibility theory and its applications: Where do we stand? In *Springer Handbook of Computational Intelligence*, pages 31–60. 2015. doi: 10.1007/978-3-662-43505-2_3.

D. Dubois and H. Prade. Practical methods for constructing possibility distributions. *International Journal of Intelligent Systems*, 31(3):215–239, 2016. doi: https://doi.org/10.1002/int.21782.

D. Dubois, J. Lang, and H. Prade. *Possibilistic Logic*, page 439–513. Oxford University Press, Inc., USA, 1994. ISBN 0198537476.

M. Elkano, J. A. Sanz, E. Barrenechea, H. Bustince, and M. Galar. CFM-BD: A distributed rule induction algorithm for building compact fuzzy models in big data classification problems. *IEEE Transactions on Fuzzy Systems*, 28(1):163–177, jan 2020. doi: 10.1109/tfuzz.2019.2900856.

N. Fatima, A. S. Imran, Z. Kastrati, S. M. Daudpota, and A. Soomro. A systematic literature review on text generation using deep neural network models. *IEEE Access*, 10:53490–53503, 2022. doi: 10.1109/access.2022.3174108.

D. Ferrucci. Introduction to "This is Watson". *IBM Journal of Research and Development*, 56:1:1–1:15, 05 2012. doi: 10.1147/JRD.2012.2184356.

E. Frank, M. A. Hall, G. Holmes, R. Kirkby, B. Pfahringer, and I. H. Witten. *Weka: A machine learning workbench for data mining.*, pages 1305–1314. Springer, Berlin, 2005. URL `http://researchcommons.waikato.ac.nz/handle/10289/1497`.

M. Frazier and L. Pitt. Learning from entailment: An application to propositional Horn sentences. In *ICML*, pages 120–127. Morgan Kaufmann, 1993a.

M. Frazier and L. Pitt. Learning from entailment: An application to propositional Horn sentences. In *ICML*, 1993b. doi: 10.1007/3-540-49730-7_11.

M. Frazier and L. Pitt. Classic learning. *Machine Learning*, 25(2-3):151–193, 1996.

S. García, J. Luengo, J. A. Sáez, V. López, and F. Herrera. A survey of discretization techniques: Taxonomy and empirical analysis in supervised learning. *IEEE Transactions on Knowledge and Data Engineering*, 25(4):734–750, 2013. doi: 10.1109/TKDE.2012.35.

C. Glanois, Z. Jiang, X. Feng, P. Weng, M. Zimmer, D. Li, W. Liu, and J. Hao. Neuro-symbolic hierarchical rule induction. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 7583–7615. PMLR, 2022. URL `https://proceedings.mlr.press/v162/glanois22a.html`.

I. J. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, Cambridge, MA, USA, 2016. `http://www.deeplearningbook.org`.

J. Guan and V. R. Lesser. On probabilistic logic. In *AI and Cognitive Science '89*, pages 113–131, London, 1990. Springer London. ISBN 978-1-4471-3164-9.

M. Hermo and A. Ozaki. Exact learning: On the boundary between Horn and CNF. *ACM Trans. Comput. Theory*, 12(1):4:1–4:25, 2020.

S. Hölldobler, S. Möhle, and A. Tigunova. Lessons learned from AlphaGo. In *YSIP2*, pages 92–101. CEUR-WS.org, 2017.

A. Horn. On sentences which are true of direct unions of algebras. *The Journal of Symbolic Logic*, 16(1):14–21, 1951. ISSN 00224812. doi: 10.2307/2268661.

J. Hühn and E. Hüllermeier. FURIA: an algorithm for unordered fuzzy rule induction. *Data Mining and Knowledge Discovery*, 19(3):293–319, Dec 2009. ISSN 1573-756X. doi: 10.1007/s10618-009-0131-8.

L. Hyafil and R. L. Rivest. Constructing optimal binary decision trees is NP-complete. *Information Processing Letters*, 5(1):15–17, 1976. ISSN 0020-0190. doi: https://doi.org/10.1016/0020-0190(76)90095-8. URL `https://www.sciencedirect.com/science/article/pii/0020019076900958`.

E. T. Jaynes. *Probability theory: The logic of science*. Cambridge University Press, Cambridge, 2003.

F. V. Jensen. *Bayesian Networks and Decision Graphs*. Statistics for Engineering and Information Science. Springer, 2001.

A. Jøsang. A logic for uncertain probabilities. *Int. J. Uncertain. Fuzziness Knowl. Based Syst.*, 9(3):279–212, 2001.

C. Joslyn. Towards an empirical semantics of possibility through maximum uncertainty. 1991.

M. J. Kearns and U. V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994. ISBN 978-0-262-11193-5. URL `https://mitpress.mit.edu/books/introduction-computational-learning-theory`.

K. Kirkpatrick. It's not the algorithm, it's the data. *Communications of the ACM*, 60: 21 – 23, 2017.

B. Konev, C. Lutz, A. Ozaki, and F. Wolter. Exact learning of lightweight description logic ontologies. In *KR*. AAAI Press, 2014.

B. Konev, A. Ozaki, and F. Wolter. A model for learning description logic ontologies based on exact learning. In *AAAI*, pages 1008–1015, 2016.

B. Konev, C. Lutz, A. Ozaki, and F. Wolter. Exact learning of lightweight description logic ontologies. *Journal of Machine Learning Research*, 18(201):1–63, 2018.

R. Kusters, Y. Kim, M. Collery, C. de Sainte Marie, and S. Gupta. Differentiable rule induction with learned relational features. Jan. 2022.

O. Kuzelka, J. Davis, and S. Schockaert. Encoding Markov logic networks in possibilistic logic. In *UAI*, pages 454–463. AUAI Press, 2015.

O. Kuzelka, J. Davis, and S. Schockaert. Learning possibilistic logic theories from default rules. In *IJCAI*, pages 1167–1173. IJCAI/AAAI Press, 2016.

O. Kuzelka, J. Davis, and S. Schockaert. Induction of interpretable possibilistic logic theories from relational data. In *IJCAI*, pages 1153–1159. ijcai.org, 2017.

J. Lang. *Possibilistic Logic: Complexity and Algorithms*, pages 179–220. Springer Netherlands, Dordrecht, 2000. ISBN 978-94-017-1737-3. doi: 10.1007/978-94-017-1737-3_5. URL `https://doi.org/10.1007/978-94-017-1737-3_5`.

V. Lavín Puente. Learning a subclass of k-quasi-Horn formulas with membership queries. *Information Processing Letters*, 111(11):550–555, 2011. ISSN 0020-0190. doi: https://doi.org/10.1016/j.ipl.2011.03.008.

Q. V. Le, M. Ranzato, R. Monga, M. Devin, G. Corrado, K. Chen, J. Dean, and A. Y. Ng. Building high-level features using large scale unsupervised learning. In *ICML*. icml.cc / Omnipress, 2012. doi: 10.48550/arXiv.1112.6209.

J. Lehmann. Dl-learner: Learning concepts in description logics. *J. Mach. Learn. Res.*, 10:2639–2642, 2009.

J. Lehmann and J. Völker. *Perspectives on Ontology Learning*, volume 18 of *Studies on the Semantic Web*. IOS Press, 2014.

R. Liaw, E. Liang, R. Nishihara, P. Moritz, J. E. Gonzalez, and I. Stoica. Tune: A research platform for distributed model selection and training. *arXiv preprint arXiv:1807.05118*, 2018.

M. McGough. How bad is Sacramento's air, exactly? Google results appear at odds with reality, some say, 2018. URL `https://www.sacbee.com/news/state/california/fires/article216227775.html`.

S. Minaee, Y. Y. Boykov, F. Porikli, A. J. Plaza, N. Kehtarnavaz, and D. Terzopoulos. Image segmentation using deep learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2021. doi: 10.1109/tpami.2021.3059968.

M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of Machine Learning*. Adaptive computation and machine learning. MIT Press, 2012.

J. Montiel et al. Scikit-multiflow: A multi-output streaming framework. *Journal of Machine Learning Research*, 19(72):1–5, 2018. doi: 10.48550/arXiv.1807.04662.

S. Muggleton. Inverse entailment and Progol. *New Generation Computing*, 13(3):245–286, Dec 1995. ISSN 1882-7055. doi: 10.1007/BF03037227. URL https://doi.org/10.1007/BF03037227.

S. Muggleton, L. De Raedt, D. Poole, I. Bratko, P. Flach, K. Inoue, and A. Srinivasan. ILP turns 20. *Machine Learning*, 86(1):3–23, Jan 2012. ISSN 1573-0565. doi: 10.1007/s10994-011-5259-2. URL https://doi.org/10.1007/s10994-011-5259-2.

N. J. Nilsson. Probabilistic logic. *Artificial Intelligence*, 28(1):71–87, 1986. ISSN 0004-3702. doi: https://doi.org/10.1016/0004-3702(86)90031-7. URL https://www.sciencedirect.com/science/article/pii/0004370286900317.

T. Okudono et al. Weighted automata extraction from recurrent neural networks via regression on state spaces. In *AAAI*, pages 5306–5314. AAAI Press, 2020. doi: 0.1609/aaai.v34i04.5977.

A. Ozaki. Learning description logic ontologies: Five approaches. Where do they stand? *KI - Künstliche Intelligenz*, 04 2020a. doi: 10.1007/s13218-020-00656-9.

A. Ozaki. On the complexity of learning description logic ontologies. In *Reasoning Web*, volume 12258 of *Lecture Notes in Computer Science*, pages 36–52. Springer, 2020b.

A. Ozaki, C. Persia, and A. Mazzullo. Learning query inseparable ELH ontologies. In *AAAI*, pages 2959–2966. AAAI Press, 2020.

S. Parsons and A. Hunter. *A Review of Uncertainty Handling Formalisms*, pages 8–37. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988. ISBN 1558604790.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

F. J. Pelletier. Metamathematics of fuzzy logic. Trends in logic, vol. 4. Kluwer Academic Publishers, Dordrecht, Boston, and London, 1998, viii 297 pp. *Bulletin of Symbolic Logic*, 6(3):342–346, 2000. doi: 10.2307/421060.

L. Peng, W. Qing, and G. Yujia. Study on comparison of discretization methods. *2009 International Conference on Artificial Intelligence and Computational Intelligence*, 4: 380–384, 2009.

C. Persia and R. Guimarães. RIDDLE: Rule induction with deep learning. In *NLDL*. septentrio.uit.no, 2023.

C. Persia and A. Ozaki. On the learnability of possibilistic theories. In *IJCAI*, pages 1870–1876. ijcai.org, 2020.

C. Persia and A. Ozaki. Extracting Horn theories from neural networks with queries and counterexamples. In *KR4HI*. CEUR workshop proceedings, 2022.

C. Persia, J. Jøsang, and A. Ozaki. Extracting rules from neural networks with partial interpretations. In *NLDL*. septentrio.uit.no, 2022.

V. Podgorelec, P. Kokol, B. Stiglic, and I. Rozman. Decision trees: An overview and their use in medicine. *Journal of Medical Systems*, 26(5):445–463, Oct 2002. ISSN 1573-689X. doi: 10.1023/A:1016409317640. URL `https://doi.org/10.1023/A:1016409317640`.

H. Prade and M. Serrurier. Bipolar version space learning. *International Journal of Intelligent Systems*, 23(10):1135–1152, 2008.

L. Qiao, W. Wang, and B. Lin. Learning accurate and interpretable decision rule sets from neural networks. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 4303–4311. AAAI Press, 2021. URL `https://ojs.aaai.org/index.php/AAAI/article/view/16555`.

J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993. ISBN 1-55860-238-0.

L. D. Raedt. Logical settings for concept-learning. *Artif. Intell.*, 95(1):187–201, 1997.

A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever. Zero-shot text-to-image generation, 2021. URL `https://arxiv.org/abs/2102.12092`.

C. Reddy and P. Tadepalli. Learning first-order acyclic Horn programs from entailment. In *ICML*, pages 472–480. Morgan Kaufmann, 1998a.

C. Reddy and P. Tadepalli. Learning first-order acyclic Horn programs from entailment. *ILP*, pages 23–37, 1998b.

S. Ross. *A First Course in Probability*. Pearson Education, Incorporated, 2014. ISBN 9780321794772. URL https://books.google.no/books?id=7yqBNAEACAAJ.

C. Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nat. Mach. Intell.*, 1(5):206–215, 2019. doi: 10.1038/s42256-019-0048-x.

M. S. Santos et al. A new cluster-based oversampling method for improving survival prediction of hepatocellular carcinoma patients. *Journal of Biomedical Informatics*, 58:49–59, 2015. ISSN 1532-0464. doi: 10.1016/j.jbi.2015.09.012.

G. Scala, A. Federico, V. Fortino, D. Greco, and B. Majello. Knowledge generation with rule induction in cancer omics. *International Journal of Molecular Sciences*, 21(1):18, dec 2019. doi: 10.3390/ijms21010018.

J. Selman and A. Fern. Learning first-order definite theories via object-based queries. In *ECML/PKDD (3)*, volume 6913 of *Lecture Notes in Computer Science*, pages 159–174. Springer, 2011.

M. Serrurier and H. Prade. Introducing possibilistic logic in ILP for dealing with exceptions. *Artif. Intell.*, 171(16-17):939–950, 2007.

G. Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, 1976. ISBN 9780691100425. URL http://www.jstor.org/stable/j.ctv10vm1qb.

G. Shafer. A mathematical theory of evidence turns 40. *International Journal of Approximate Reasoning*, 79:7–25, 2016. ISSN 0888-613X. doi: https://doi.org/10.1016/j.ijar.2016.07.009. URL https://www.sciencedirect.com/science/article/pii/S0888613X16301141. 40 years of Research on Dempster-Shafer Theory.

S. Shalev-Shwartz and S. Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

M. A. Sheikh, A. K. Goel, and T. Kumar. An approach for prediction of loan approval using machine learning algorithm. In *2020 International Conference on Electronics and Sustainable Communication Systems (ICESC)*, pages 490–494, 2020. doi: 10.1109/ICESC48915.2020.9155614.

A. Shih, A. Darwiche, and A. Choi. Verifying binarized neural networks by Angluin-style learning. In *SAT*, 2019. doi: 10.1007/978-3-030-24258-9_25.

H. Shindo, M. Nishino, and A. Yamamoto. Differentiable inductive logic programming for structured examples. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(6):5034–5041, May 2021. doi: 10.1609/aaai.v35i6.16637. URL https://ojs.aaai.org/index.php/AAAI/article/view/16637.

M. Sipser. *Introduction to the theory of computation*. PWS Publishing Company, 1997.

J. F. Torres, D. Hadjout, A. Sebaa, F. Martínez-Álvarez, and A. Troncoso. Deep learning for time series forecasting: A survey. *Big Data*, 9(1):3–21, 2021. doi: 10.1089/big.2020.0159. URL https://doi.org/10.1089/big.2020.0159. PMID: 33275484.

L. G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, 1984. ISSN 0001-0782.

V. Vapnik and A. Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. 1971. doi: 10.1007/978-3-319-21852-6_3.

J. Vreeken, M. van Leeuwen, and A. Siebes. Krimp: mining itemsets that compress. *Data Mining and Knowledge Discovery*, 23(1):169–214, Jul 2011. ISSN 1573-756X. doi: 10.1007/s10618-010-0202-x.

P. Walley. *Statistical Reasoning with Imprecise Probabilities*. Chapman & Hall, 1991.

P. Walley. Measures of uncertainty in expert systems. *Artificial Intelligence*, 83(1):1–58, 1996. ISSN 0004-3702. doi: https://doi.org/10.1016/0004-3702(95)00009-7. URL https://www.sciencedirect.com/science/article/pii/0004370295000097.

P. Walley and W. Peter. *Statistical Reasoning with Imprecise Probabilities*. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. Taylor & Francis, 1991. ISBN 9780412286605. URL https://books.google.no/books?id=-hbvAAAAMAAJ.

O. Watanabe. A formal study of learning via queries. In M. S. Paterson, editor, *Automata, Languages and Programming*, pages 139–152, Berlin, Heidelberg, 1990. Springer Berlin Heidelberg. ISBN 978-3-540-47159-2.

G. Weiss, Y. Goldberg, and E. Yahav. Extracting automata from recurrent neural networks using queries and counterexamples. In *ICML*, volume 80, pages 5244–5253. PMLR, 2018. doi: 10.48550/arXiv.1711.09576.

G. Weiss, Y. Goldberg, and E. Yahav. Learning deterministic weighted automata with queries and counterexamples. In *NeurIPS*, 2019.

R. Wexler. When a computer program keeps you in jail: how computers are harming criminal justice., 2017. URL https://www.nytimes.com/2017/06/13/opinion/how-computers-are-harming-criminal-justice.html.

L. WJ. A rule-based process control method with feedback. *Advances in Instrumentation 41, 169–175*, 1987.

W. Wong, W. Liu, and M. Bennamoun. Ontology learning from text: A look back and into the future. *ACM Comput. Surv.*, 44(4):20:1–20:36, 2012.

P. Xu, Z. Ding, and M. Pan. A hybrid interpretable credit card users default prediction model based on RIPPER. *Concurrency and Computation: Practice and Experience*, 30(23):e4445, feb 2018. doi: 10.1002/cpe.4445.

R. R. Yager and L. Liu, editors. *Classic Works of the Dempster-Shafer Theory of Belief Functions*, volume 219 of *Studies in Fuzziness and Soft Computing*. Springer, 2008.

L. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338–353, 1965. ISSN 0019-9958. doi: https://doi.org/10.1016/S0019-9958(65)90241-X. URL `https://www.sciencedirect.com/science/article/pii/S001999586590241X`.

L. Zadeh. Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*, 1(1):3–28, 1978. ISSN 0165-0114. doi: https://doi.org/10.1016/0165-0114(78)90029-5. URL `https://www.sciencedirect.com/science/article/pii/0165011478900295`.

Y. Zhang, P. Tiňo, A. Leonardis, and K. Tang. A survey on neural network interpretability. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 5 (5):726–742, 2021. doi: 10.1109/TETCI.2021.3100641.

Graphic design: Communication Division, UiB / Print: Skipnes Kommunikasjon AS

uib.no

ISBN: 9788230854259 (print)
9788230867600 (PDF)