Learning in Order to Reason

Dan Roth

TR-02-95
January 1995

Center for Research in Computing Technology
Harvard University
Cambridge, Massachusetts
Learning in Order to Reason

A thesis presented
by
Dan Roth
to
The Division of Applied Sciences
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy
in the subject of
Computer Science

Harvard University
Cambridge, Massachusetts
January, 1995
Abstract

Any theory aimed at understanding commonsense reasoning, the process that humans use to cope with the mundane but complex aspects of the world in evaluating everyday situations, should account for its flexibility, its adaptability, and the speed with which it is performed.

In this thesis we analyze current theories of reasoning and argue that they do not satisfy those requirements. We then proceed to develop a new framework for the study of reasoning, in which a learning component has a principal role. We show that our framework efficiently supports a lot “more reasoning” than traditional approaches and at the same time matches our expectations of plausible patterns of reasoning in cases where other theories do not.

In the first part of this thesis we present a computational study of the knowledge-based system approach, the generally accepted framework for reasoning in intelligent systems. We present a comprehensive study of several methods used in approximate reasoning as well as some reasoning techniques that use approximations in an effort to avoid computational difficulties. We show that these are even harder computationally than exact reasoning tasks. What is more surprising is that, as we show, even the approximate versions of these approximate reasoning tasks are intractable, and these severe hardness results on approximate reasoning hold even for very restricted knowledge representations.

Motivated by these computational considerations we argue that a central question to consider, if we want to develop computational models for commonsense reasoning, is how the intelligent system acquires its knowledge and how this process of interaction with its environment influences the performance of the reasoning system. The Learning to Reason framework developed and studied in the rest of the thesis exhibits the role of inductive learning in achieving efficient reasoning, and the importance of studying reasoning and learning phenomena together. The framework is defined in a way that is intended to overcome the main computational difficulties in the traditional treatment of reasoning, and indeed, we exhibit several positive results that do not hold in the traditional setting. We develop Learning to Reason algorithms for classes of theories for which no efficient reasoning algorithm exists when represented as a traditional (formula-based) knowledge base. We also exhibit Learning to Reason algorithms for a class of theories that is not known to be learnable in the traditional sense. Many of our results rely on the theory of model-based representations that we develop in this thesis. In this representation, the knowledge base is represented as a set of models (satisfying assignments) rather than a logical formula. We show that in many cases reasoning with a model-based representation is more efficient than reasoning with a formula-based representation and, more significantly, that it suggests a new view of reasoning, and in particular, of logical reasoning.

In the final part of this thesis, we address another fundamental criticism of the knowledge-based system approach. We suggest a new approach for the study of the non-monotonicity of human commonsense reasoning, within the Learning to Reason framework. The theory developed is shown to support efficient reasoning with incomplete information, and to avoid many of the representational problems which existing default reasoning formalisms face.

We show how the various reasoning tasks we discuss in this thesis relate to each other and conclude that they are all supported together naturally.
Table of Contents

1 Introduction
   1.1 Techniques ................................................. 7
   1.2 Outline of the Thesis ...................................... 8

2 Hardness of Approximate Reasoning ........................................ 12
   2.1 The Computational Complexity of Counting Problems ............... 13
   2.2 Summary of Model-Counting Results ............................. 15
      2.2.1 Statements of Results .................................... 17
   2.3 Reducing Approximate Reasoning to Counting ...................... 18
      2.3.1 Degree of Belief ......................................... 18
      2.3.2 Bayesian Belief Networks ................................. 19
      2.3.3 Reasoning with Approximate Theories ....................... 23
      2.3.4 Constraint Satisfaction Problems ......................... 26
      2.3.5 Related Results .......................................... 28
   2.4 Concluding Remarks .......................................... 28
   2.5 Proofs ................................................................ 29
      2.5.1 Preliminaries .............................................. 29
      2.5.2 Exact Counting ........................................... 30
      2.5.3 Approximate Counting ................................... 31
      2.5.4 Positive Results ........................................... 33

3 Learning to Reason: The Framework .......................................... 43
   3.1 Preliminaries: Learning and Reasoning ............................. 47
      3.1.1 Reasoning ................................................ 47
      3.1.2 Learning to Classify ..................................... 48
   3.2 Motivation: A Sampling Approach ................................ 51
   3.3 Learning to Reason: Definitions .................................. 54
   3.4 The relations between L2R and L2C ................................ 55
3.5 I2R by Combining Learning and Reasoning .......................... 56
3.5.1 Learning to Reason via PAC Learning .......................... 56
3.5.2 Learning to Reason via Mistake Bound Learning .............. 57
3.6 Concluding Remarks .............................................. 58

4 Reasoning with Models ............................................. 60
4.1 Monotone Theory ................................................. 63
4.2 Deduction with Models ........................................... 66
  4.2.1 Exact Deduction ............................................. 66
  4.2.2 Exact Deduction with Approximate Theories ............... 68
4.3 Horn Theories .................................................... 70
  4.3.1 Characteristic Models ....................................... 71
  4.3.2 General Queries ............................................ 72
4.4 The Size of Model Based Representations ......................... 73
4.5 Applications ...................................................... 75
  4.5.1 Languages with a Small Basis ............................... 76
  4.5.2 Main Applications .......................................... 77
  4.5.3 Structure Identification .................................... 78
4.6 Abduction with Models .......................................... 79
4.7 Robustness of Model-Based Representations ...................... 81
4.8 Concluding Remarks ............................................. 83

5 Learning to Reason: Deduction ..................................... 85
5.1 Learning to Reason without Reasoning .......................... 86
  5.1.1 Model-Based Representation ................................ 86
  5.1.2 Horn Queries ............................................... 87
5.2 Learning to Reason without Learning to Classify ............... 87
  5.2.1 Exact Learning to Reason ................................... 88
  5.2.2 PAC Learning to Reason .................................... 89
5.3 Concluding Remarks ............................................. 92

6 Learning to Reason with Incomplete Information .................. 93
6.1 First Attempt .................................................... 96
  6.1.1 Learning to Reason without Reasoning: The Partial Assign- 
        ments Case .................................................. 97
  6.1.2 Conclusions ............................................... 100
6.2 Handling Incomplete Information: The Framework ............... 101
6.3 Default Reasoning ............................................... 105
  6.3.1 Relation to Default Reasoning Formalisms .................. 111
6.4 Learning to Reason ............................................. 112
  6.4.1 Learning to Classify over \{0, 1, *\}^n ..................... 112
6.5 General Reasoning Tasks ....................................... 114
  6.5.1 Answering General Queries ................................ 114
  6.5.2 Relation to Model-Based Reasoning ....................... 117
## List of Figures

2.1 Complexity of (Approximately) Counting Satisfying Assignments . . . 16
2.2 The Belief Network generated from the 2MONCNF formula \( \Sigma \) . . . 22
2.3 Counting independent sets of a tree . . . . . . . . . . . . . . . . . . . . . 41
3.1 S-Reason: A Sampling approach to Reasoning . . . . . . . . . . . . . 52
4.1 MBR: Model-Based Reasoning . . . . . . . . . . . . . . . . . . . . . . . . . 61
4.2 \( d \)-MBR: Model-Based Reasoning within Context . . . . . . . . . . . 82
5.1 The Algorithm \( Ex-L2R-DNF \) . . . . . . . . . . . . . . . . . . . . . . . . . 88
5.2 The Procedure \( Find-min-model(x, b) \) . . . . . . . . . . . . . . . . . . . 89
5.3 The Algorithm \( PAC-L2R-DNF \) . . . . . . . . . . . . . . . . . . . . . . . . 90
6.1 LAZY-MBR: Model-Based Reasoning with Partial Assignments . . . 99
Acknowledgments

I am deeply grateful to my advisor Les Valiant. My graduate career could not have been as enriching and enjoyable as it was if it were not for Les’s guidance and support. Les was always available to listen to my ideas and share his ideas with me. His insightful comments and his work have been an inspiration for my work. It has been a great pleasure to have Les as my advisor.

I thank my family - my wife Michal whose love and support have kept us going and enjoying these past few years, my daughter Noam, who started graduate school as a baby and is now a complete and wonderful human being, for being such an inspiration, and finally to Ido, who joined us recently.

Special thanks to Roni Khardon, who collaborated with me on a large part of this thesis. This research could not have been accomplished without his contribution and working with him has been a great pleasure.

I thanks the many colleagues and friends who have helped make my graduate school experience so great. Special thanks go to Karen Daniels, Eyal Kushilevitz and Marios Mavronicolas, with whom I have collaborated on other papers, and to my officemate for the past two years Scott Decatur. I thank them for their friendship, their help and everything I have learned from them.

I have had valuable conversations with many other colleagues during the course of my research and at the risk of omitting some if them I would like to thank Avrim Blum, Nader Bshouty, Rina Dechter, Russ Greiner, Haym Hirsh, Henri Kautz, Mike Kearns, Lenny Pitt, Greg Provan, and Bart Selman. I thank David Mumford, Michael Rabin and Stuart Shieber, who served on my thesis committee.

Many thanks to Carol Harlow and Pauline Mitchell for their help in all the little things that are so important in making this place a pleasant environment to work in.

This thesis is dedicated to the memory of my mother, the most influential person in my life.

This research was supported by NSF grants CCR-89-02500 and CCR-92-00884 and by DARPA AFOSR-F4962-92-J-0466.
Bibliographic Notes

Much of the research presented in this thesis has appeared elsewhere, in one way or another. Chapter 2 is based on the paper “The hardness of approximate reasoning” (to appear in the Artificial Intelligence Journal). An extended abstract of this paper appeared in the Proceedings of the International Joint Conference of Artificial Intelligence, 1993 (Roth, 1993).

Chapters 3 and 5 are based on the paper “Learning to Reason”, a joint work with Roni Khardon. An extended abstract of this paper appeared in the Proceedings of the National Conference on Artificial Intelligence, 1994 (Khardon and Roth, 1994b).

Chapter 4, with the exception of few sections, is based on the paper “Reasoning with Models”, a joint work with Roni Khardon. An extended abstract of this paper appeared in the Proceedings of the National Conference on Artificial Intelligence, 1994 (Khardon and Roth, 1994e).

Chapter 6 is based on the paper “Learning to Reason with Incomplete Information” (Roth, 1994).
Any theory aimed at understanding *commonsense* reasoning, the process that humans use to cope with the mundane but complex aspects of the world in evaluating everyday situations, should account for its flexibility, its adaptability and the speed with which it is performed.

Consider, for example, the task of language understanding, which humans perform effortlessly and effectively. It depends upon our ability to disambiguate word meanings, recognize speaker’s plans, perform predictions and generate explanations. These, and other “high level” cognitive tasks such as high level vision and planning have been widely interpreted as *inference* tasks and collectively comprise what we call commonsense reasoning.

To act sensibly in the world an intelligent agent must know about the world and must be able to use her\(^1\) knowledge effectively. Based on this and on “our introspective views of our own mental structure”, as stated by McCarthy and Hayes (1969), a research program, the knowledge-based system approach (McCarthy, 1958), was launched and is today the generally accepted framework for reasoning in intelligent systems. The idea is to store the knowledge in some *representation language* with a well defined meaning assigned to its sentences. The sentences are stored in a Knowledge Base \((KB)\) which is combined with a reasoning mechanism that can be used to determine what can be inferred from the sentences in the \(KB\). There are many knowledge representations that can be used to represent the knowledge in a knowledge-based system. Different representation systems (e.g., a set of logical rules, a probabilistic network) are associated with corresponding reasoning mechanisms, each with its own merits and range of applications. Given a logical knowledge base, for example, reasoning can be abstracted, as a deduction task of determining whether a sentence, assumed to capture the situation at hand, is logically implied

\(^1\)No implication with respect to gender is intended. Accordingly, when referring to some abstract person, we shall sometimes use she, to mean ‘she or he’.

1
by the knowledge base. In all cases, the emphasis of this approach is on comprehensibility (McCarthy and Hayes, 1969; Pearl, 1988): knowledge should be encoded so that it is readily accessible.

The question of how this knowledge might be acquired and whether this should influence how the performance of the reasoning system is measured is not usually considered. As a matter of fact, in a review paper for a special issue of the Artificial Intelligence Journal dedicated to the foundations of AI (Kirsh, 1991), it is stated that many schools of research in Artificial Intelligence consider the principle "Learning can be added later" as one of their fundamental abstract assumptions.

Computational considerations, however, render this self-contained approach to reasoning inadequate for commonsense reasoning. We argue this in the first part of this thesis, in which we adopt the theory of computational complexity as our methodology for approaching the understanding of reasoning tasks. This methodology can be used, for example, to identify knowledge representations with which one can reason efficiently, or even to reject a formalization of a cognitive task such as reasoning, if we can prove that the claimed, precisely defined task, cannot be realized with what we consider a reasonable amount of resources (e.g., number of time steps). Thus this methodology can guide us in the search for a better way to formalize and study this task.

Following the seminal work of Brachman and Levesque (1984) numerous researchers have investigated the computational complexity of various knowledge representations and reasoning tasks within the knowledge-based system approach. While most of this work is aimed at identifying classes of limited expressiveness with which one can perform some sort of reasoning efficiently (Brachman and Levesque, 1984; Levesque, 1992; Selman, 1990), perhaps the most significant conclusion was that most forms of reasoning, not only the task of deduction, are intractable. Of particular interest in this context are the results on default reasoning tasks (Selman, 1990; Kautz and Selman, 1991; Papadimitriou, 1991), where the increase in complexity is clearly at odds with the intuition that reasoning with defaults should somehow reduce the complexity of reasoning.

The first part of this thesis is devoted to a study within this framework. We present a comprehensive study of various methods used in approximate reasoning, as well as reasoning techniques such as constraint satisfaction and knowledge compilation, that use approximations in an effort to avoid computational difficulties. We show that these tasks are even harder computationally than exact reasoning tasks. What is more surprising is that, as we show, even the approximate versions of these approximate reasoning tasks are intractable. Moreover, we show that trying to identify classes of limited expressiveness with which one can perform some sort of reasoning is an almost hopeless task: the severe hardness results on approximate reasoning hold even for extremely restricted knowledge representations.
Studies of the computational limitations of current formalizations of reasoning have led us to doubt and challenge the traditional knowledge-based system approach to reasoning. We view the computational results developed in the first part of the thesis, along with the abundant results provided by other authors, as providing evidence to the effect that a learning component must be a principal component of a system capable of performing high level cognitive tasks. In the second part of the thesis we embark on the development of a framework in which one can evaluate this approach.

The fact that learning must take a central part in performing any high level cognitive task is evident, we believe, even without attending to detailed computational arguments. When considering the sensory system, it is reasonable to assume that there are specialized systems to deal with specific tasks. In high level cognitive tasks, the situation is different. The variety of cognitive tasks for which we are required to perform some sort of inference, from debugging a computer program to understanding a joke and cooking, cannot be executed by specialized systems. The conclusion is that the machinery behind high level cognitive tasks cannot be specialized to the task it is executing; instead, the system should be specialized to the task of learning how to perform these tasks.

While the central role of learning in cognition is acknowledged by many, most lines of research nevertheless study the phenomenon of “learning” separately from that of “reasoning”. In contrast to the “Learning can be added later” approach referred to above (Kirsh, 1991), we argue in this thesis that if one wants to develop computational models that account for the flexibility, adaptability and speed of reasoning, a central question to consider is how the intelligent system acquires its knowledge and how this process, of interaction with its environment, influences the performance of the reasoning system.

The Learning to Reason framework developed here highlights the role of inductive learning in achieving efficient reasoning, and illustrates the importance of studying reasoning and learning phenomena together.

In this framework the intelligent agent is given access to her favorite learning interface, and is also given a grace period in which she can interact with this interface and construct her representation of the world. Her reasoning performance is measured only after this period, when she is presented with queries that are relevant to the world, and has to answer them. The approach is aimed at overcoming the main computational difficulties in the traditional treatment of reasoning which stem from its separation from the “world”. First, by allowing the reasoning task to interact with the world (as in the known learning models), we avoid the rigid syntactic restrictions imposed in the knowledge-based systems framework on the intermediate knowledge representation. Second, we make explicit the dependence of the reasoning performance on the input from the environment. This is possible only because the agent interacts with the world when constructing her knowledge representation.
Using the Learning to Reason approach we exhibit several positive results which do not hold in the traditional setting. We develop Learning to Reason algorithms for classes of theories for which no efficient reasoning algorithm exists when represented as a traditional (formula-based) knowledge base. We also exhibit a Learning to Reason algorithm for a class of theories that is not known to be learnable in the traditional sense (which we call here Learning to Classify). The additional reasoning power of the agent is gained, as in nature, through the interaction with the world.

In developing these results we use some notions that distinguish our approach further from the traditional knowledge-based system approach. We briefly mention some of them here. The first is the notion of restricted queries. Following the pac-learning approach (Valiant, 1984) we present the view that the world is very complicated and there is no hope of acquiring an exact representation of it; our aim should be to acquire enough information with which to cope effectively in the world. Since we are interested in formalizing commonsense reasoning, we need to develop a theory that supports reasoning about complicated worlds, but does not require the reasoner to answer all the possible queries efficiently, rather than restricting the world we reason about. This should be contrasted with the common approach in the study of knowledge representations, where the goal is to identify classes of “worlds”, with limited expressiveness, which support efficient reasoning with respect to all queries. We comment that the approach of “restricting the world” has been criticized also on the grounds that existing results do not meet the strong tractability requirements for commonsense reasoning as described, for example, in (Shastri, 1993), even though (as argued, for example, in (Doyle and Patil, 1991)) the inference deals with limited expressiveness and is sometimes restricted in implausible ways. Our use of restricted queries is motivated also by computational results we present in the first part of the thesis. We show that even when one is willing to settle for an approximate representation of the world, without restricting the class of the queries any algorithm is doomed to be wrong on a vast majority of them.

The second important notion we develop here is that of a model-based representation. The assumption that the intelligent agent has to keep her knowledge in some representation and use it when she reasons is basic to this framework. We allow the reasoner, however, to choose her own representation and even to use different representations for different tasks. Many of our results rely on the theory of model-based representations that we develop in this thesis. In this representation, the agent keeps her knowledge of the world as a set of models (satisfying assignments) rather than a logical formula. While the notion of “reasoning from examples” has been alluded to previously on a qualitative basis, mostly by cognitive psychologists (Johnson-Laird, 1983; Johnson-Laird and Byrne, 1991; Kosslyn, 1983), we develop here a formal treatment for it in the context of logical reasoning.

We develop a theory of model-based representations, in which we fully characterize a set of theories for which the model-based representation is compact and
provides efficient reasoning. Computationally this is intriguing, since we can show that reasoning with a model-based representation is efficient in many cases where the formula-based representation does not support efficient reasoning. Equally important, we believe, is the view that model-based representations suggest about reasoning, and in particular, deductive reasoning. Borrowing from Brooks (1991), we show formally, though in a narrow context, that “intelligence is in the eye of the beholder”. Namely, using a model-based representation, our agent behaves logically (to an outside observer), even though her knowledge representation consists of a set of models and not a logical formula and she does not use any logic or “theorem proving”.

The traditional way to represent knowledge and abstract the reasoning problem has been criticized not only on computational but also on even more fundamental grounds, that it cannot capture what people view as plausible patterns of reasoning. In particular, as was argued first by Minsky (1975), it cannot support the non-monotonicity of reasoning. In the final part of the thesis we address this aspect of reasoning.

Indeed, it is widely acknowledged today that a large part of our everyday reasoning involves arriving at conclusions that are not entailed by our “theory” of the world. Many conclusions are derived in the absence of information that is sufficient to imply them. This type of reasoning is naturally non-monotonic: the conclusions are retractable, since further evidence may force a revision of belief.

In light of this, many researchers have tried to augment the knowledge base and to modify the inference mechanisms accordingly, to allow incomplete information. Many attempts have been made to augment the true knowledge we have about the world with a set of default rules that capture only “typical” cases, or a set of ad-hoc preference criteria. The quest is for a reasoning system that, given a query, responds in a way that agrees with what we know about the world and some subset of the default assumptions and at the same time supports our intuition about a plausible conclusion. As mentioned above, computational considerations render all the formalisms suggested for default reasoning within this framework inadequate for commonsense reasoning. Moreover, the study of reasoning in the presence of incomplete information within the knowledge-based systems framework has shown that even capturing what people view as plausible patterns of reasoning is not easy (see, e.g., (Touretzky, Hory, and Thomason, 1987), “A clash of intuitions”). Most formalisms, in order to capture some aspects of “default” reasoning, had to give up others.

In the final part of this thesis, we suggest a new approach for the study of the non-monotonicity of human commonsense reasoning, within the Learning to Reason approach. As we do throughout this thesis, at the base of this approach is the view that commonsense reasoning is an inductive phenomenon. In this final part we add the premise that missing information in the interaction of the agent with the environment may be as informative for future interactions as observed information.
We formalize this intuition and present the problem of reasoning from incomplete information as a problem of learning attribute functions ranging over a generalized domain. Our treatment of incomplete information in this part of the thesis follows the suggestion made in (Valiant, 1994b). While there, in an effort to formalize the notion of Rationality, Valiant presents a more comprehensive view of the phenomena that comprise cognition, here we present a more detailed account of reasoning in the presence of incomplete information, focusing on presenting it as a problem of Learning to Reason.

Unlike previous theories of reasoning in the presence of incomplete information, we are not interested in providing a theory of defaults, but rather a theory of inference. The reason is that we believe that one cannot evaluate whether a default is true in the world but only whether a conclusion derived by a reasoning system is acceptable. The representation suggested provides a richer language for dealing with reasoning problems. Moreover, it can be learned efficiently from interaction with the environment. The theory developed is shown to support efficient reasoning with incomplete information, and to avoid many of the representational problems which existing default reasoning formalisms face.

We show how the various reasoning tasks we discuss in this thesis relate to each other and conclude that all the reasoning tasks are supported together naturally. In order to derive at a conclusion our agent employs her inference procedures, using the knowledge representations she has learned in her interactions with the world. Whether the task looks to an outside observer as pure deduction, or as inference “with defaults” is determined mainly by what the observer knows about the world.

The Learning to Reason framework is inspired by and is similar in spirit to the Neuroidal model developed by Valiant (1994a). The model developed there provides a more comprehensive approach to cognition and, akin to our approach, it views learning as an integral and crucial part of the process. There, the reasoner reasons from a learned knowledge base, a complex circuit, and this can be modeled by our framework. Indeed, reasoning in the Neuroidal model shares many properties with the Learning to Reason framework. One difference, however, is that, in an effort to give a more formal treatment of a reasoner that has learned her knowledge base, we restrict our discussion to consistent worlds. This is not true in the final part of this thesis, where we discuss more general representations which do not force us to assume consistent worlds. There, we introduced the attribute-function representation since this may be a better starting point to a more formal investigation of these questions in other models and in particular, in Valiant’s neuroidal model.

A good methodological suggestion made by Levesque and Brachman (1987), states that

there is a big difference between a precise and predictable model of (say) sloppy reasoning, and a sloppy model of perfect (or other) reasoning and
therefore the issue of carefully and precisely characterizing the Knowledge Representation system holds equally well for both.

This advice was understood, we believe, as if the only way to develop a precise and theoretically sound theory of reasoning is by studying knowledge representations within the knowledge-based systems framework (or even within the logic approach to knowledge representation), and any other approach, not following these lines, is a “scruffy” type of approach (Birnbaum, 1991).

However, the approach developed in this thesis suggests an “operational” approach to studying reasoning, that is nevertheless rigorous and amenable to analysis. We show that this approach efficiently supports a lot “more reasoning” than traditional approaches and at the same time it matches our expectations of plausible patterns of reasoning in cases where other theories do not.

Perhaps the major difference between the knowledge-based system approach to reasoning and the Learning to Reason approach is that our approach suggests that in order to make theories of reasoning work in practice, we need to train them over a large number of examples. Therefore, finding good and large test beds on which to validate this theory is one of the most important next steps and the real test of this theory.

1.1 Techniques

This thesis uses techniques and tools from various areas of computer science, mainly theoretical computer science, computational learning theory and AI. We supply the relevant background material in each of the chapters. Here it will suffice to give a brief summary.

We use techniques from computational complexity to prove hardness results on problems of counting and approximate counting of combinatorial structures in our study of the hardness of approximate reasoning (Chapter 2). We use combinatorial tools and a combinatorial characterization of Boolean functions in developing the theory of reasoning with models (Chapter 4).

We discuss various knowledge representation formalisms and reasoning tasks throughout the thesis. Reasoning techniques such as computing degrees of belief, Bayesian belief networks, constraint satisfaction and reasoning with approximate theories are introduced and analyzed in the study of the hardness of approximate reasoning (Chapter 2). Deduction and abduction in logical knowledge bases are discussed in Chapter 4. Various default reasoning formalisms are discussed in our treatment of Learning to Reason with incomplete information (Chapter 6).

We use techniques from computational learning theory throughout the thesis. The models used are presented when we introduce the Learning to Reason framework (Chapter 3). Some well known results and techniques are used in Chapters 5 and 6.
Next we introduce some standard terminology and notation that are used throughout the thesis.

In general we shall consider problems of reasoning and learning where the “world” is modeled as a Boolean function \( f : \{0,1\}^n \rightarrow \{0,1\} \). The reasoning literature prefers the term *propositional expression* for Boolean function, and a *propositional language* for a class of Boolean functions. We use those terms interchangeably.

We consider a set \( X = \{x_1, \ldots, x_n\} \) of variables, each of which is associated with a world’s attribute and can take the value 1 or 0 to indicate whether the associated attribute is true or false in the world.

Assignments (points, vectors) are mappings from \( X \) to \( \{0,1\} \), and we treat them as points in \( \{0,1\}^n \) with the natural mapping. Assignments in \( \{0,1\}^n \) are denoted by \( x, y, z \), and weight(\( x \)) denotes the number of 1 bits in the assignment \( x \). A literal is either a variable \( x_i \) (called a positive literal) or its negation \( \overline{x_i} \) (a negative literal). A clause is a disjunction of literals, and a CNF formula is a conjunction of clauses. For example \( (x_1 \lor \overline{x_2}) \land (x_3 \lor \overline{x_1} \lor x_4) \) is a CNF formula with two clauses. A term is a conjunction of literals, and a DNF formula is a disjunction of terms. For example \( (x_1 \land \overline{x_2}) \lor (x_3 \land \overline{x_1} \land x_4) \) is a DNF formula with two terms. A CNF formula is Horn if every clause in it has at most one positive literal. We note that every Boolean function has many possible representations, and in particular both a CNF representation and a DNF representation. The size of a CNF and a DNF representation is the number of clauses or the number of terms in the representation, respectively.

An assignment \( x \in \{0,1\}^n \) satisfies \( f \) if \( f(x) = 1 \). Such an assignment \( x \) is also called a model of \( f \). If \( f \) is a theory of the “world”, a satisfying assignment of \( f \) is sometimes called a possible world. By “\( f \) implies \( g \)”, denoted \( f \models g \), we mean that every model of \( f \) is also a model of \( g \). Throughout the thesis, when no confusion can arise, we identify a Boolean function \( f \) with the set of its models, namely \( f^{-1}(1) \). Observe that the connective “implies” (\( \models \)) used between Boolean functions is equivalent to the connective “subset or equal” (\( \subseteq \)) used for subsets of \( \{0,1\}^n \). That is, \( f \models g \) if and only if \( f \subseteq g \).

### 1.2 Outline of the Thesis

#### The Hardness of Reasoning

Investigating the computational cost of tasks that are of interest to AI has been argued (Levesque, 1986; Valiant, 1984) to be essential to understanding and characterizing these tasks and to finding knowledge representation systems adequate for them.

In Chapter 2 we consider various methods used in approximate reasoning such as computing degrees of belief and Bayesian belief networks, as well as reasoning techniques such as constraint satisfaction and knowledge compilation, that use ap-
proximation to avoid computational difficulties, and reduce them to model-counting problems over a propositional domain. We then show that these problems are computationally intractable even in surprisingly restricted cases and even if we settle for an approximate solution.

In particular, we prove that counting satisfying assignments of propositional languages is intractable even for Horn and monotone formulae, and even when the size of clauses and number of occurrences of the variables are extremely limited. This should be contrasted with the case of deductive reasoning, where Horn theories and theories with binary clauses are distinguished by the existence of linear time satisfiability algorithms. What is even more surprising is that, as we show, even approximating the number of satisfying assignments (i.e., “approximating” approximate reasoning), is intractable for most of these restricted theories. For some of the remaining restricted classes of propositional formulae we develop efficient algorithms that count their satisfying assignments exactly.

We show that the approach of counting solutions is valuable also when evaluating other reasoning techniques. In particular, we use it to argue that even when willing to settle for reasoning with approximate theories, no general purpose algorithm can support correct reasoning with respect to all queries.

The negative results proved in this chapter provide the main motivation for many of the ideas developed later in this thesis.

Learning to Reason: The Framework

In Chapter 3 we introduce a new framework for the study of reasoning. The Learning (in order) to Reason approach developed here combines the interfaces to the world used by known learning models with the reasoning task and a performance criterion suitable for it. We motivate the approach by changing the reasoning scenario considered in Chapter 2: we allow the agent to interact with the environment and investigate the effect it has on the efficiency of reasoning. We then proceed to define the framework in a way that is aimed at overcoming the main computational difficulties in the traditional treatment of reasoning, which stem from its separation from the “world”. We define an interface to the world, in the spirit of the known learning models, and a performance criteria that make explicit the dependence of the reasoning performance on the input from the environment. We show how previous results from learning theory and reasoning fit into this framework, and how they can be used to obtain Learning to Reason algorithms. We also illustrate the current disconnection between the fields of learning and reasoning by showing that in many cases positive results in one field cannot be used in the other.

Reasoning with Models

In Chapter 3 we develop a model-based approach to reasoning, in which the knowledge base is represented as a set of models (satisfying assignments) rather than a logical formula, and the set of queries is restricted. We show that for every propo-
sitional knowledge base \((KB)\) there exists a set of characteristic models with the property that a query is true in \(KB\) if and only if it is satisfied by the models in this set. We argue that using a model-based representation side-steps many of the difficulties introduced by the traditional formula-based representation. To support this we fully characterize a set of theories for which the model-based representation is compact and provides efficient reasoning and show that these include cases where the formula-based representation does not support efficient reasoning. In addition, we consider the model-based approach to abductive reasoning and show that for any propositional \(KB\), reasoning with its model-based representation yields an abductive explanation in time that is polynomial in its size.

We give application of the theory to other problems in reasoning, and in particular show that the theory developed here generalizes the model-based approach to reasoning with Horn theories (Kautz, Kearns, and Selman, 1993), and captures even the notion of reasoning with Horn-approximations (Selman and Kautz, 1991). Finally, we discuss some robustness issues of model-based representations and show how they support an incremental approach to reasoning.

**Learning to Reason: Deduction**

In Chapter 5 we exhibit the computational advantages of the Learning to Reason approach.

We prove new results that do not hold in the traditional setting. First, we develop Learning to Reason algorithms for a class of propositional languages for which there are no efficient reasoning algorithms, when represented as a traditional (formula-based) knowledge base. In line with the Learning to Reason framework, these algorithms, that aim at different tasks (i.e., classes of queries), use different intermediate knowledge representations. Second, we exhibit Learning to Reason algorithms for a class of propositional languages that is not known to be learnable in the traditional sense.

**Learning to Reason: Incomplete Information**

In Chapter 6 we suggest a new approach for the study of the non-monotonicity of human commonsense reasoning within the Learning to Reason framework. In addition to the premise, assumed throughout the thesis, that commonsense reasoning is an inductive phenomena, we present here a view of incomplete information: missing information in the interaction of the agent with the environment may be as informative for future interactions as observed information. We formalize this intuition and present the problem of reasoning from incomplete information as a problem of learning attribute functions ranging over the domain \(\{0, 1, *\}^n\).

We consider examples that illustrate various aspects of the non-monotonic reasoning phenomena that have been used over the years as benchmarks for various formalisms. We translate them into Learning to Reason problems in order to exhibit the advantages of our approach. The theory developed is shown to support efficient
reasoning with incomplete information, and to avoid many of the representational problems which existing default reasoning formalisms face.

Our approach to reasoning with incomplete information is then put in the context of the rest of the work presented in this thesis, to show that all the reasoning tasks are supported together naturally.

We conclude by discussing some interesting theoretical and empirical issues this thesis raises.
The real-time constraint on cognition, although metaphorically a curse to the organism, is a genuine blessing to the theorist.

Allen Newell, Unified Theories of Cognition

Investigating the computational cost of tasks that are of interest to AI has been argued (Levesque, 1986; Valiant, 1984) to be essential to understanding and characterizing these tasks, and to finding knowledge representation systems adequate for them. The problem discussed most extensively in this context is the problem of propositional satisfiability, the typical NP-hard problem, which is of special concern to AI because of its direct relationship to deductive reasoning. Many other forms of reasoning, including default reasoning, planning and others which make direct appeal to satisfiability, have also been shown to be NP-hard. In practice, there is a fundamental disagreement about the implications of these results. There is no debate, however, that something has to be given up: restrict the form of the statements in the knowledge base, settle for approximate output and so on. One product of the research in this direction has been the identification of restricted languages for which propositional satisfiability can be solved efficiently (e.g., Horn theories).

In this chapter we consider a related problem, that of counting satisfying assignments of propositional formulae. We argue that the role of satisfiability problems in AI problems in which deduction is of special concern, is replaced by counting satisfying assignments in problems which use approximate reasoning techniques. To support this argument we show that various methods used in approximate reasoning, such as computing degrees of belief and Bayesian belief networks, are equivalent, computationally, to solving counting problems. We also show that the approach of counting solutions is valuable when evaluating techniques such as constraint satisfi-
faction and knowledge compilation, that use approximations in an effort to avoid computational difficulties.

We analyze the computational complexity of counting satisfying assignments of propositional languages. We prove that this is intractable even for Horn and monotone formulae, and even when the size of clauses and number of occurrences of a variable in the formula are extremely limited. This should be contrasted with the case of deductive reasoning, where Horn theories and theories with binary clauses are distinguished by the existence of linear time algorithms for their satisfiability. What is more surprising is that, as we show, even approximating the number of satisfying assignments (that is, “approximating” approximate reasoning), is intractable for most of those restricted theories. For some of the remaining restricted classes of propositional formulae we develop efficient algorithms that count their satisfying assignments exactly.

While our positive results can sometimes be used to find tractable classes for the approximate reasoning technique discussed, we believe that the main contribution of this chapter is the presentation of these widely applicable and surprising hardness results. This implies that research should be directed away from investigating structural constraints of the “world” and towards investigating other constraints, and suggests that we reconsider how we model the reasoning problem. We discuss these issues further in Section 2.4, where we summarize this chapter and use the results to motivate the work presented in the rest of the thesis.

In the next section we give background material from the computational complexity of counting problems, and in Section 2.2 we present our positive and negative results on exact and approximate counting of satisfying assignments. The main results are presented in Section 2.3, where we put the model-counting results in the context of various approximate reasoning techniques, by reducing those techniques to counting problems. Proofs of the technical results appear in Section 2.5.

### 2.1 The Computational Complexity of Counting Problems

We give in this section a brief overview of the computational complexity of counting problems and the related problems of approximate counting and random generation of solutions. For a detailed discussion consult (Valiant, 1979a; Valiant, 1979b; Garey and Johnson, 1979; Jerrum, Valiant, and Vazirani, 1986).

With a large number of decision problems we can naturally associate a counting problem. For example, counting the number of satisfying assignments of a Boolean formula, counting the number of perfect matchings in a bipartite graph and counting the number of cycles in a graph. Clearly, the counting version is at least as hard as the decision problem but in many cases, even when the decision problem is easy, no computationally efficient method is known for counting the number of distinct solutions. The class \#P was introduced by Valiant (1979a; 1979b) in an
effort to explain these phenomena. A counting problem belongs to \( \#P \) if there is a non-deterministic algorithm such that for any instance \( I \) of the associated decision problem, the number of “guesses” that lead to acceptance of \( I \) is equal to the number of distinct solutions to \( I \), and such that the algorithm is polynomial in the size of \( I \). As usual, the hardest problems in the complexity class are \textit{complete} in the class.

In particular, it was shown that counting the number of satisfying assignments of a CNF formula as well as the counting versions of many other NP-complete problems are complete for \( \#P \). More significantly, it was also shown that the counting versions of many problems in P are also complete for the same class. Examples of the latter include counting the number of satisfying assignments of a DNF formula, counting the number of cycles in a graph and many other problems (Valiant, 1979a; Valiant, 1979b; Provan and Ball, 1983).

Problems that are \( \#P \)-complete are at least as hard as NP-complete problems, but probably much harder. Evidence to the hardness of problems in \( \#P \) is supplied by a result of Toda (1989) which implies that one call to a \( \#P \) oracle suffices to solve any problem in the polynomial hierarchy in deterministic polynomial time. This may serve also as an indication that \( \#P \) is outside of the polynomial hierarchy. It is therefore natural to consider the problem of approximate counting. The notion of approximation we use is that of \textit{relative approximation} (Karp and Luby, 1983; Stockmeyer, 1985; Jerrum, Valiant, and Vazirani, 1986). Let \( M, M' \) be two positive numbers and \( \delta \geq 0 \). We say that \( M' \) \textit{approximates} \( M \) within \( \delta \) when

\[
\frac{M'}{(1 + \delta)} \leq M \leq M'(1 + \delta).
\]

Indeed, approximating a solution to a \( \#P \) problem might be easier than finding an exact solution. In fact, it is no harder than solving NP-hard problems (Stockmeyer, 1985). For example, there exists a polynomial time randomized algorithm that approximates the number of satisfying assignments of a DNF formula within any constant ratio (Karp and Luby, 1983; Jerrum, Valiant, and Vazirani, 1986). It is possible, though, for a \( \#P \)-complete problem, even if its underlying decision problem is easy, to resist even an efficient \textit{approximate solution}. An example for that was given in (Jerrum, Valiant, and Vazirani, 1986), and in this chapter we exhibit further examples of this phenomenon. We prove, for various propositional languages for which solving satisfiability is easy, that it is NP-hard to approximate the number of satisfying assignments even in a very weak sense.

We use the notion of relative approximation to discuss probabilities as well. It is worth noticing therefore that this notion of approximation is preserved when computing ratios of quantities.

In the following we say that two approximation algorithms \( A \) and \( B \) are \textit{polynomially related} when \( A \), a relative approximation algorithm for the probability \( P \) that is polynomial in \( \delta \), can be simulated in polynomial time to yield \( B \), which relatively approximates \( P' \), and is also polynomial in \( \delta \), and vice versa.

\footnote{In (Valiant, 1979a) the definition is given in terms of “counting Turing machines”.
}
Assume we can approximate $M_1$ and $M_2$ to within $\delta$. That is, we can find $M_1', M_2'$ such that $M_1'/(1+\delta) \leq M_1 \leq (1+\delta)M_1'$ and $M_2'/(1+\delta) \leq M_2 \leq (1+\delta)M_2'$. Then,

$$\frac{M_1}{M_2} / (1+\delta) \leq \frac{M_1}{M_2} \leq (1+\delta)^2 \frac{M_1'}{M_2'}.$$

Thus, this yields a relative approximation of the ratio $\frac{M_1}{M_2}$ as well (within $2\delta + \delta^2$). In particular, when computing the conditional probability $P(Y = y | X = x)$, of the event $Y = y$ given evidence $X = x$, since

$$P(Y = y | X = x) = \frac{P(Y = y, X = x)}{P(X = x)}$$

we conclude that:

**Proposition 2.1.1** The complexity of computing a relative approximation to the conditional probability $P(Y = y | X = x)$ is polynomially related to that of computing a relative approximation to the unconditional probability $P(Y = y)$.

We note that a related class of problems of interest to AI, that of randomly generating solutions from a uniform distribution, was shown in (Jerrum, Valiant, and Vazirani, 1986) to be equivalent to randomized approximate counting, for a wide class of problems. (All natural problems considered here, e.g., finding satisfying assignments of Boolean formulae and various graph problems are in this class.) It is therefore enough, from the computational complexity point of view, to consider the problems of exact and approximate counting, as we do here.

### 2.2 Summary of Model-Counting Results

In this section we summarize our results on exact and approximate counting of satisfying assignments of propositional languages. Those include hardness results for exact and approximate counting and positive results for exact counting. Complete proofs of the results are given in Section 2.5.

Let $\#(SAT, \mathcal{L})$ ($\#(SAT, \mathcal{L})$) denote the problem of counting (approximating, respectively) the number of satisfying assignments of a given formula from the propositional language $\mathcal{L}$. Given the problem $\#(SAT, \mathcal{L})$, a problem hierarchy is obtained whenever we place additional restrictions or relaxations on the allowed instances. Given propositional languages $\mathcal{L}_1$ and $\mathcal{L}_2$, define $\mathcal{L}_1 \subseteq \mathcal{L}_2$ if every instance of $\mathcal{L}_1$ is also an instance of $\mathcal{L}_2$. (e.g., HORN $\subseteq$ CNF.) Clearly, if we can solve the problem $\#(SAT, \mathcal{L}_2)$ we can solve the problem $\#(SAT, \mathcal{L}_1)$; to prove hardness results it is therefore enough to consider the most restricted languages. The same argument holds for the corresponding approximation problem.

We use the following notations and conventions in denoting propositional languages: if LANG is a class of Boolean formulae and $k, l$ are integers, then $k\text{LANG}$ denotes the subclass of formulae in LANG in which a clause consists of up to $k$
Figure 2.1: Complexity of (Approximately) Counting Satisfying Assignments

literals; $\mu$-LANG denotes the class of all LANG formulae in which no variable occurs more than $l$ times. $l$ is the degree of the formulae. For example, 2MONCNF consists of monotone CNF formulae with clauses of length 2; 3$\mu$-2HORN consists of Horn formulae with clauses of length 2 in which no variable occurs more than 3 times.

SAT: Any Boolean formulae.

MON: Boolean formulae in which all variables are unnegated (monotone formulae).

CNF: Boolean formulae in Conjunctive Normal Form

MONCNF: Monotone CNF.

HORN: A CNF in which clauses have up to one unnegated variable (Horn clauses).

2BPMONCNF: A 2MONCNF in which the set of variables can be divided into two sets, and every clause contains one variable from each.

Acyclic-2MONCNF: Given a 2MONCNF theory $\phi$, let $G$ be an undirected graph containing a vertex for every variable in $\phi$ and an edge connecting two vertices if and only if the corresponding variables appear in the same clause. $\phi$ is Acyclic-2MONCNF if this corresponding graph is acyclic.

Acyclic-2HORN: Given a 2HORN theory $\phi$, let $G$ be a directed graph containing a vertex for every literal in $\phi$ and an edge from every vertex corresponding to a
literal in the body of a rule (i.e., negative variable in the clause representation of the rule) to the vertex corresponding to a literal in the head of a rule (i.e., positive variable in the clause representation of the rule). Two special nodes $T$ and $F$ are added for clauses with empty body or empty head. $\phi$ is Acyclic-2HORN if every connected component of this corresponding graph is a directed tree.

Figure 2.1 summarizes our results; it presents a hierarchy of propositional languages along with a classification according to the complexity of $\#(\text{SAT, } \mathcal{L})$ and $\#(\text{SAT, } \mathcal{L})$. Based on the above comment these results imply similar results on other, less restricted languages.

It can be seen that for various propositional languages having efficient algorithms for satisfiability, and even for very restricted versions of these (e.g., 3$\mu$-2HORN), exact counting is complete for $\#P$. In fact, for the case of Horn theories, the situation is fully understood, and we give an efficient algorithm for the only possible case, 2$\mu$-2HORN. The situation for approximate counting is even more surprising: for very restricted classes of Horn theories (e.g., 3$\mu$-2HORN) it is NP-hard to approximate the number of satisfying assignments even within, say, $2^{2\pi n}$ (for formulae over $n$ variables). Similar results hold for 2MONCNF theories, for which the bounded degree case is open. Our positive results virtually complete the complexity picture and can be directly applied in some of the reasoning techniques considered.

### 2.2.1 Statements of Results

We now formally state the technical results outlined above. We state the results only for some of the important languages; results for other languages can be easily deduced by inclusion, as described above. Proofs are given in the Section 2.5.

**Theorem 2.2.1 [Hardness of Exact Counting]** Let $\Sigma \in \mathcal{L}$ be a propositional formula on $n$ variables. If $\mathcal{L}$ is one of the following propositional languages, counting the number of satisfying assignments of $\Sigma$ is complete for $\#P$:

1. $\mathcal{L} = 2\text{MONCNF}$ (Valiant, 1979b)
2. $\mathcal{L} = 2\text{BPMONCNF}$ (Provan and Ball, 1983)
3. $\mathcal{L} = 2\text{HORN}$
4. $\mathcal{L} = 3\mu$-2HORN
5. $\mathcal{L} = 4\mu$-2MON

**Theorem 2.2.2 [Hardness of Approximation]** Let $\Sigma \in \mathcal{L}$ be a propositional formula on $n$ variables, and let $\epsilon > 0$ be any constant. If $\mathcal{L}$ is one of the following

---

2Notice that we place the language 2Horn above 2MONCNF even though 2HORN does not contain 2MONCNF. 2HORN contains, however, 2anti-MONCNF (were all the variables in each formula are negated rather than unnegated) and thus, clearly, all the counting results that hold for 2MONCNF hold also for 2anti-MONCNF.
propositional languages, approximating the number of satisfying assignments of $\Sigma$ to within $2^{\frac{1}{2}n}$ is NP-hard:

(1) $\mathcal{L} = 2\mu$-MONCNF
(2) $\mathcal{L} = 3\mu$-HORN

**Theorem 2.2.3** [Positive Results] Let $\Sigma \in \mathcal{L}$ be a propositional formula on $n$ variables. If $\mathcal{L}$ is one of the following propositional languages, then there exists an efficient algorithm for counting the number of satisfying assignments of $\Sigma$.

(1) $\mathcal{L} = 2\mu$-MONCNF
(2) $\mathcal{L} = 2\mu$-HORN
(3) $\mathcal{L} = \text{Acyclic-2MONCNF}$
(4) $\mathcal{L} = \text{Acyclic-2HORN}$

### 2.3 Reducing Approximate Reasoning to Counting

In this section we consider various techniques for approximate reasoning and show that in each case inference is equivalent to solving a counting problem. We start by considering the case of computing degree of belief, the underpinning of approximate reasoning. We then consider Bayesian belief networks, reasoning with approximate theories and constraint satisfaction problems. Finally, we discuss some previous work that relates approximate reasoning to counting problems, for which our results here also apply.

#### 2.3.1 Degree of Belief

The inference of a degree of belief is a generalization of deductive inference which can be used when the knowledge base is augmented by, e.g., statistical information, or in an effort to avoid the computationally hard task of deductive inference.

Consider a Knowledge Base consisting of a propositional theory $\Sigma$ and assume we would like to assign a *degree of belief* to a particular statement $\alpha$. This situation is natural in various AI problems such as planning, expert systems and others, where the actions an agent takes may depend crucially on this degree of belief. Nilsson (1986) suggests that the kind of reasoning used in expert system is the following: “we are given a knowledge base of facts (possibly, with their associated probabilities); we want to compute the probability of some sentence of interest. ... According to probabilistic logic, the probability of a sentence is the sum of the probabilities of the sets of possible worlds in which that sentence is true... ”

Indeed, the general approach to computing degree of belief is that of assigning equal degree of belief to all basic “situations” consistent with the knowledge base, and computing the fraction of those which are consistent with the query. Much work has been done on how to apply this principle, and how to determine what are the basic situations (Carnap, 1950; Bacchus, 1990; Bacchus et al., 1992).
We consider here the question of computing the degree of belief in a restricted case, in which the knowledge base consists of a propositional theory and contains no statistical information. The hardness results we get in this restricted case just highlight the computational difficulties in the more general cases. Using the above approach, all possible models of the theory are given equal weight and we are interested in the computational complexity of computing the degree of belief of a propositional formula, that is, the fraction of models that are consistent with a propositional query.

Given a propositional theory $\Sigma$ on $n$ variables, the probability that $\Sigma$ is satisfied, $P_\Sigma$, is computed over the uniform distribution over the set $\{0, 1\}^n$ of all possible assignments of $\Sigma$.

$$P_\Sigma = \text{Prob}(\Sigma \equiv T) = \frac{|M(\Sigma)|}{2^n},$$

where $M(\Sigma)$ denotes the set of all satisfying assignments of $\Sigma$, and $|M(\Sigma)|$ denotes its size, and $T$ stands for the Truth value.

Given a propositional theory $\Sigma$ and a propositional statement $\alpha$, the degree of belief in $\alpha$, is the conditional probability of $\alpha$ with respect to $\Sigma$, $P_{\alpha|\Sigma}$, that is, the fraction of satisfying assignments of $\Sigma$ that satisfy $\alpha$:

$$P_{\alpha|\Sigma} = \text{Prob}(\alpha \land \Sigma \equiv T | \Sigma \equiv T) = \frac{|M(\alpha \land \Sigma)|}{|M(\Sigma)|}.$$

The observation that $|M(\alpha)| = 2^n \cdot P_{\alpha\land\neg p}$ for any variable $p$, together with the discussion in Section 2.1 (Proposition 2.1.1) implies:

**Proposition 2.3.1** The complexity of computing (approximating) the degree of belief in a propositional statement with respect to a propositional theory, is polynomially related to the complexity of computing (approximating) the number of models of a propositional statement.

Together with the results in Theorem 2.2.1 and Theorem 2.2.2 we have:

**Theorem 2.3.2** The problem of computing the degree of belief in a propositional statement (over $n$ variables) with respect to a propositional theory, is complete for $\#P$. For every fixed $\epsilon > 0$, approximating this probability within $2^{n^{1-\epsilon}}$ is NP-hard.

### 2.3.2 Bayesian Belief Networks

Bayesian belief networks provide a natural method for representing probabilistic dependencies among a set of variables and are considered an efficient and expres-
sive language for representing knowledge in many domains (Holtzman, 1989). We consider here the class of multiple connected belief network, i.e., networks that contain at least one pair of nodes (variables) that have more than one undirected path connecting them. It has been argued that the expressiveness of these networks is required for representing knowledge in several domains, like medicine. We first present briefly a general class of belief networks and the associated inference problem and then show how to reduce the inference problem to that of counting satisfying assignments of a propositional formula. For definitions and an elaborate discussion of Bayesian belief networks, the expressiveness of this representation, and the type of inference one can utilize using it, see (Pearl, 1988).

A Bayesian belief network (causal network) consists of a graphical structure augmented by a set of probabilities. The graphical structure is a directed acyclic graph (DAG) in which nodes represent random variables (domain variables) and edges represent the existence of direct causal influence between the linked variables. A conditional probability is associated with the group of edges directed toward each node (and not with each single edge). Prior probabilities are assigned to source nodes (i.e., any node without any incoming edge). We represent the belief network as a triple \((V, E, P)\), where \(V\) is the set of vertices (variables), \(E\) is the set of edges and \(P\) is the set of probabilities. In particular, \(P\) consists of prior probability functions, \(P(X_i = x_i)\), for every source node \(X_i\) and conditional probabilities functions, \(\{P(X_i|Y_{ij})\}_{Y_{ij} \in p_{X_i}}\), for each node \(X_i\) with a set \(p_{X_i}\) of direct predecessors. (We use the notation \(P(X_i = x_i)\) and \(P(x_i)\) interchangeably, when it is clear that we refer to the variable \(X_i\).) See the construction in the proof of Theorem 2.3.3 for an example of a belief network. In general, not every probability distribution can be represented by a Bayesian belief network. However, given a DAG it is easy to specify consistently the conditional probabilities. One needs only to make sure that the conditional probabilities attached satisfy, for every node \(X_i\), \(\sum_{x_i} P(X_i = x_i|p_{X_i}) = 1\).

For complexity analysis, we take our complexity parameter to be \(n\), the number of nodes in the belief network. Notice that the conditional probabilities tables associated with the network might be exponential in \(n\). Practitioners of Bayesian belief networks try to avoid this case, of course. In our reduction the conditional probabilities tables will have concise representations, polynomial in the number of nodes in the network, and in this sense one can think of our complexity measure as if it is the total size of network, including the conditional probabilities tables.

The general inference problem using belief networks is that of calculating the posterior probability \(P(S_1|S_2)\), where \(S_1 (S_2)\) is either a single instantiated variable or a conjunction of instantiated variables. The most restricted form of probabilistic inference, determining \(P(Y = T)\) for some propositional variable \(Y\) (with no explicit conditioning information), was analyzed by Cooper (1990) who proved that it is NP-hard. This hardness result for the exact inference problem shows that one cannot expect to develop general-purpose algorithms for probabilistic inference that have a polynomial running time and therefore there is a need to divert attention toward trying to construct approximation algorithms for probabilistic inference. Our results
show that this is not the case:

**Theorem 2.3.3** The problem of computing the probability that a node in a Bayesian belief network is true is complete for \( \#P \). Moreover, for every fixed \( \epsilon > 0 \), approximating this probability within \( 2^{n^{1-\epsilon}} \) (where \( n \) is the size of the network) is NP-hard.

**Proof:** The proof is based on the reduction from (Cooper, 1990). The two major differences are that (1) we reduce the problem of counting satisfying assignments of a propositional formula to that of computing the probability that a node in a Bayesian belief network is true, and (2) based on the results from Section 2.2 we can start our reduction from a restricted propositional formula, yielding a more restricted network topology.

In the following we reduce the problem of counting satisfying assignments of a 2MONCNF\(^5\) formula to that of computing the probability that a node in a Bayesian belief network is true. Since our reduction preserves the number of satisfying assignments this reduction holds for the problem of approximating the probability as well.

Consider an instance of 2MONCNF, \( \Sigma = \{ c_1, c_2, \ldots, c_m \} \) where \( c_i \) are clauses on a set \( U = \{ u_1, u_2, \ldots, u_n \} \) of \( n \) Boolean variables. We construct a belief network \( BN = (V, E, P) \) containing variable \( Y \) such that

\[
2^n P(Y = T) = |M(\Sigma)|.
\]

To construct \( BN = (V, E, P) \) we show how to define the vertices \( V \), the edges \( E \) and the set of prior and conditional probabilities \( P \). \( V \) is defined to be a set of \( n + m + 1 \) vertices, one for each variable \( u_i \), one for every clause \( c_j \) and one for \( Y \). The set of edges \( E \) consists of up to \( 3m \) edges: a variable \( u_i \) is connected to all clauses \( c_j \) in which it appears (i.e., total of up to \( 2m \) edges, since \( \Sigma \in 2\text{MONCNF} \)); \( Y \) is connected to all clauses \( c_j \). Figure 2.2 depicts the \( BN \) generated using the above procedure for the instance of 2MONCNF in which \( U = \{ u_1, u_2, u_3, u_4 \} \), and

\[
\Sigma = \{ (u_1 \lor u_2), (u_1 \lor u_3), (u_2 \lor u_4) \}.
\]

The set of probabilities \( P \) is defined as follows: Each of the source nodes \( u_i \), \( 1 \leq i \leq n \) is given a prior probability of \( 1/2 \) to be \( T \). For incoming edges to the node corresponding to the clause \( c_j \) we define the conditional probability such that the node becomes \( T \) only when it is satisfied by the assignment to the variables in the clause. Formally, if \( c_j = u_{j1} \lor u_{j2} \) (\( 1 \leq j \leq m \)), define the conditional probabilities by:

\[
P(c_j = T | u_{j1}^1 = v_1, u_{j2}^2 = v_2) = \begin{cases} 1 & \text{if the assignment } u_{j1}^1 = v_1, u_{j2}^2 = v_2 \text{ satisfies } c_j \\ 0 & \text{otherwise} \end{cases}
\]

\(^5\)This is not possible in (Cooper, 1990), since the results there hinge on the hardness of solving satisfiability, which can be done in polynomial time for 2MONCNF.
Finally, the conditional probability for the edges coming into the node $Y$ is defined by

$$P(Y = T | c_1, c_2, \ldots c_m) = \begin{cases} 1 & \text{if } c_1 = T, c_2 = T, \ldots, c_m = T \\ 0 & \text{otherwise} \end{cases}$$

It is easy to see that the structure $(V, E, P)$ defined is indeed a Bayesian belief network. Also, it is clear that the construction of a belief network from a given 2MONCNF instance can be accomplished in time that is polynomial in the size of the 2MONCNF instance\(^6\).

We now compute the probability $P(Y = T)$. Let $u = (u_1, \ldots, u_n)$ be an assignment of the $n$ input variables (that is, $u \in \{0, 1\}^n$), and $c = (c_1, \ldots, c_m)$ be an assignment of the $m$ clauses (that is, $c \in \{0, 1\}^m$).

By the construction above we then have that

$$P(Y = T) = \sum_{s=0}^{2^n-1} \sum_{t=0}^{2^m-1} P(Y = T | c = t)P(c = t | u = s)P(u = s). \quad (2.1)$$

Suppose $\Sigma$ is satisfiable, and let $s_1, s_2, \ldots, s_k$ be the satisfying assignments. Clearly, for $i = 1, \ldots, k$, $P(u = s_i) = \frac{1}{2^n}$. Also, by the definition of the conditional probability for $i = 1, \ldots, k$, we have that $P(c = c(s_i) | u = s_i) = P(Y = T | c = c(s_i)) = 1$, and for any other assignment, these terms are equal to 0. Thus, the internal sum in Equation 2.1 is equal to $\frac{1}{2^n}$ when $s$ is a satisfying assignment of $\Sigma$, and is equal\(^6\)This relies on the fact that we can define the conditional probabilities concisely. In general, every node can be associated with a conditional probability table that is exponential in the size of the network.
to 0 otherwise. We get,
\[ P(Y = T) = \frac{|M(\Sigma)|}{2^n}. \]

Applying now the results in Theorem 2.2.1 and Theorem 2.2.2 completes the proof.

We have considered the computational complexity of computing the probability of a node in a Bayesian belief network being true. To compute a conditional probability, that is, \( P(Y = y|X = x) \), where \( X, Y \) might be sets of nodes in the network, we notice that
\[ P(Y = y|X = x) = \frac{P(Y = y, X = x)}{P(X = x)}. \]

It is clear that exact computation of the conditional probability is as hard as computing the unconditional probability (taking, e.g., \( X \) to be a single source node). Based on Proposition 2.1.1 this is the case also for the problem of approximating the conditional probability, and therefore we can conclude:

**Theorem 2.3.4** The problem of computing the conditional probability of a node given evidence in a Bayesian belief network, is complete for \( \#P \). Moreover, for every fixed \( \epsilon > 0 \), approximating this probability within \( 2^{n-1-\epsilon} \) (where \( n \) is the size of the network) is NP-hard.

Finally we note that as in (Cooper, 1990), this reduction can be modified to hold for restricted network topology (limited in-degree, out-degree, etc.) Further restrictions to the topologies of the network can be utilized if we reduce problems of counting satisfying assignments of syntactically restricted CNF formulae to that of computing the probability that a node in the network is true. In light of the results in Section 2.2, this can yield even stronger hardness results. Recently, Dagum and Luby (1993) presented an even stronger result, implying the hardness of computing an absolute approximation of probabilities in Bayesian networks. The results here are different in that we show that the inference is equivalent to counting, and combined with the results in Section 2.2, it implies hardness results even for restricted network topologies.

### 2.3.3 Reasoning with Approximate Theories

The theory of reasoning with *approximate theories* was introduced by Selman and Kautz in a series of papers (Selman and Kautz, 1991; Kautz and Selman, 1991; Kautz and Selman, 1992) as a new approach to developing efficient knowledge representation systems.

The goal is to speed up inference by replacing the original theory by two theories that belong to a different propositional language \( \mathcal{L} \) and approximate the original theory. One approximate theory implies the original theory (a lower bound) and the other one is implied by it (an upper bound). While reasoning with the approximations instead of the original theory does not guarantee exact reasoning, it
can sometimes provide a quick (but not necessarily complete, see below) answer to the inference problem. This can happen when $L$ allows for efficient deduction, e.g., if $L$ is the class of propositional Horn formulae. Of course, computing the approximations is a hard computational problem, and this is why it is suggested as an “off-line” compilation process. Some computational aspects of computing theory approximations and reasoning with them are studied also in (Cadoli, 1993; Greiner and Schuurmans, 1992), and a general characterization of reasoning with approximate theories is discussed in Chapter 4. In the following we concentrate on discussing Horn approximation.

For notational convenience, when no confusion can arise, we identify in this section the propositional theory $\Sigma$ with the set of its models (satisfying assignments). Observe that the connective “implies” ($\models$) used between Boolean functions (propositional formulae) is equivalent to the connective “subset or equal” ($\subseteq$) used for subsets of models. That is, $\Sigma_1 \models \Sigma_2$ if and only if $\Sigma_1 \subseteq \Sigma_2$.

Consider a propositional theory $\Sigma$. The Horn theories $\Sigma_{lb}$, $\Sigma_{ub}$ are a Horn lower-bound and Horn upper-bound, respectively, of $\Sigma$, if and only if

$$\Sigma_{lb} \models \Sigma \models \Sigma_{ub}$$

or, equivalently, in subset notations,

$$\Sigma_{lb} \subseteq \Sigma \subseteq \Sigma_{ub}.$$  

$\Sigma_{glb}$ and $\Sigma_{lub}$, the greatest Horn lower-bound and least Horn upper-bound, respectively, of $\Sigma$, are called Horn approximations of the original theory $\Sigma$.

In order to answer $\Sigma \models \alpha$, we use a Horn approximation based inference procedure in the following way: (1) test if $\Sigma_{ub} \models \alpha$. If the answer to (1) is “yes”, then the inference procedure answers “yes”, $\Sigma \models \alpha$. Otherwise, (2) test if $\Sigma_{glb} \models \alpha$. If the answer to (2) is “no”, then the inference procedure answers “no”, $\Sigma \not\models \alpha$. Otherwise, the inference procedure returns “don’t know”.

Aside from the two computational problems related to Horn approximations, namely, computing the approximations and the question of the size of the formula representing the approximations (See, e.g., (Selman and Kautz, 1991; Kautz and Selman, 1991; Kautz and Selman, 1992; Cadoli, 1993) and the discussion in Chapter 4.) a third major question, that is harder to analyze, is the question of evaluating the utility of reasoning with the approximate theories. Clearly, if for a given query we have either $\Sigma_{lb} \models \alpha$ or $\Sigma_{glb} \not\models \alpha$, the answer to the question $\Sigma \models \alpha$ is correct. The total performance of the inference procedure is determined, though, by how many queries are answered “don’t know”, forcing the procedure to resort to the original inference problem in order to answer the query.

Consider a theory $\Sigma$, and let $\Sigma_{lub}$ be its least upper bound approximation.

---

7The implication problem for Horn theories can be solved in linear time in the combined length of the theory and the query. This remains true for even a broader class of queries such as DNF formulae where each disjunct contains at most one negative literal and arbitrary CNF formulae.
Proposition 2.3.5 The number of queries for which the reasoning with approximate theories returns “don’t know” is at least exponential in $|\Sigma_{lb} \setminus \Sigma|$.

Proof: Let $S = \Sigma_{lb} \setminus \Sigma$. For every subset $s \subseteq S$ define the query $\alpha_s = \Sigma \cup s$. (That is, the set of models of $\alpha$ consists of the models of $\Sigma$ and the models in the set $s$.) Then, for all $s \subseteq S$, $\Sigma_{glb} \models \alpha_s$ (since $\Sigma_{glb} \models \Sigma$), and $\Sigma_{lb} \not\models \alpha_s$. Therefore, for all the $2^{|s|} - 1$ queries $\alpha_s$, reasoning with approximate theories returns “don’t know”.

In (Kautz, Kearns, and Selman, 1994) it is shown that, for a family of propositional languages $\mathcal{L}$ which consists of $k$-Horn formulae (all Horn formulae with up to $k$ literals in a clause), one can construct examples of theories $\Sigma$ for which $|\Sigma_{lb} \setminus \Sigma|$ is exponential in the number of variables, where $\Sigma_{lb}$ is the least upper bound of $\Sigma$ in $\mathcal{L}$. (Surprisingly, one can even construct $(k+1)$-Horn theories with these properties). Using Proposition 2.3.5, this leads to a double exponential number of queries for which reasoning with approximate theories returns “don’t know”. In Chapter 4 tools are developed that allow for a construction of such examples for every language $\mathcal{L}$ with respect to which we want to consider theory approximation. Here we just briefly describe one example, for the case of Horn approximation:

Consider the theory

$$\Sigma = (x_1 \lor x_2) \land (x_3 \lor x_4) \land \ldots \land (x_{n-1} \lor x_n).$$

It is easy to see that the number of models of $\Sigma$ is $3^{n/2}$. However, the least upper approximation of $\Sigma$ with respect to Horn, $\Sigma_{lb}$, can be shown to contain all the models in $\{0, 1\}^n$, that is, it is of size $2^n$. This can be argued from the fact that the set of models of any Horn formula is closed under intersection (bitwise “and”) (see, e.g., (Dechter and Pearl, 1992)). Therefore, the size of $\Sigma_{lb} \setminus \Sigma$ is exponential in the number $n$ of variables.

This question is partially addressed in (Greiner and Schuurmans, 1992), where learning techniques are used to find a locally-optimal approximation. However, in (Greiner and Schuurmans, 1992), as is done in general in the theory of Horn approximations, an approximation is defined in terms of containment, (that is, logical strength), and there is no guarantee that this approximation is “close” to the optimal one, nor that the optimal one approximates the original theory within any reasonable bound, in the sense that it answers some fraction of the queries correctly.

Taking the approach of counting solutions, as we suggest in this chapter, can shed some light on this problem. As argued above, a more reasonable way to estimate the utility of reasoning with approximate theories is to define it in terms of proximity in the number of models, since this correlates positively with the number of queries answered correctly (i.e., not answered with “don’t know”) by the approximations. The examples presented indicate that, when no restriction are imposed on the queries, even syntactically similar propositional languages (e.g., $(k+1)$-Horn
formulæ and kHorn formulæ) can be far enough in terms of the number of models to produce unacceptable behavior.

The argument presented above, as exhibited in Proposition 2.3.5, shows that the heart of the problem is the fact that the queries presented to the reasoner are unrestricted. Thus it motivates an investigation in the direction of reasoning with restricted queries, where it might be possible to avoid these difficulties. Indeed, in (Kautz and Selman, 1994) an experimental analysis is presented in which, under severe restrictions on the classes of queries allowed, reasoning with approximate theories is shown to succeed on a large percentage of the queries. In Chapter 4 a general analysis is developed and it is shown, in particular, that reasoning with the least upper bound of a theory $\Sigma$ with respect to $L$ is always correct if and only if the queries belong to the language $L$.

2.3.4 Constraint Satisfaction Problems

Constraint satisfaction problems (CSP) provide a convenient way of expressing declarative knowledge, by focusing on local relationships among entities in the domain.

A constraint satisfaction problem (Dechter, 1992) involves a set of $n$ variables $x_1, \ldots, x_n$ having domains $D_1, \ldots, D_n$, where each $D_i$ defines the set of values that variable $x_i$ may assume. An n-ary relation on these variables is a subset of the Cartesian product $D_1 \times D_2 \times \cdots \times D_n$. A constraint between variables is a subset of the Cartesian product of their domains. The set of all n-tuples satisfying all the constraints are the solutions to the CSP. The problem is either to find all the solutions, or one solution.

In general, constraint satisfaction problems are hard, as a generalization of satisfiability of CNF formulæ. A CNF formula is a constraint satisfaction problem in which $D_i = \{0, 1\}$ for each $i$, and each clause is the relation consisting of all the tuples for which at least one literal is 1. The set of solutions of the CSP is the set of satisfying assignments of the formula. In particular, if we consider a network of binary constraints over $D_i = \{0, 1\}$, as is usually done, the problem can be represented as a 2SAT formula.

Finding all the solutions is clearly an enumeration problem, and based on the results in Section 2.2, even the associated counting problem is $\#P$-complete for almost all non-trivial cases.

However, even finding one solution of a constraint satisfaction problem is known to be hard in general, as discussed above, and different heuristics techniques have

---

8Not every n-ary relation can be represented by a network of binary constraints with n-variables (Montanari, 1974).

9We comment, though, that Valiant's results ((Valiant, 1979b), Fact 7) imply that under simple conditions (e.g., when finding one solution is easy and the problem satisfies a form of self-reducibility), enumerating the solutions is polynomial in their number even when the counting problem is hard. These conditions trivially hold for Horn formulæ, and therefore for subclasses of CSP as well.
been used to find approximate solutions. To discuss those under the counting point of view presented here, we first observe that the above discussion implies that computing (approximating) the number of solutions of a (binary) CSP problem is at least as hard as computing (approximating) the number of solutions of a 2M0NCNF formula.

Together with the results in Theorem 2.2.1 and Theorem 2.2.2 we have:

**Theorem 2.3.6** The problem of computing the number of solutions of a (binary) CSP problem is complete for \#P. For every fixed \( \epsilon > 0 \), approximating this number within \( 2^{n^{1-\epsilon}} \) is NP-hard.

Search techniques were traditionally used to solve CSPs. These techniques require, in the worst case, exponential search time, and analyzing those techniques in order to get some performance guarantees is usually hard.

We exemplify how the counting point of view taken here can be used to evaluate one class of heuristics (Dechter and Pearl, 1988) and restrict its feasibility. Dechter and Pearl suggest to use counting to guide the search according to an estimate of the confidence we have that a specific solution can be extended further to a full solution.

More specifically, it is suggested to (1) simplify the pending subproblems (i.e., make some simplifying assumptions about the continuing portion of the search graph), (2) count the number of consistent solutions in each simplified subproblem, and (3) order the candidates according to these counts and use them to decide among the pending subproblems.

The intuition behind the heuristics is that “the number of solutions found in the simplified version of the problem is a good estimate to the number of solutions in the original problem and thus is indicative of the chance to retain at least one surviving solution.”

The results we present in Section 2.2 therefore restrict the utility of these heuristics in two ways: First, the simplified subproblem must be a tree, or another syntactically constrained structure (see Theorem 2.2.3), in order to be able to get a count of the solutions of the simplified version. More significantly, in case the original problem possesses a non-trivial structure, the number of solutions of the simplified version is not indicative at all of the number of solutions of the original problem. If it were, this could be used to approximate the number of solutions of the original problem, which we have shown to be hard. To summarize:

**Corollary 2.3.7** Using counting heuristics to constraint satisfaction problem is computationally intractable.

On the other hand, the positive results (Theorem 2.2.3) can be used to identify restricted domains for which these counting techniques can be shown useful.
2.3.5 Related Results

The most related result to the work presented here is a reduction of yet another approximate reasoning technique to a counting problem. Orponen (1990) shows, by reduction, that the problem of computing Dempster’s rule of combination, the main tool in the Dempster-Shafer theory of evidence is at least as hard as the problem of computing the number of satisfying assignments of a propositional CNF formula\(^{10}\). Those results can now be strengthened using Theorem 2.2.1 and Theorem 2.2.2. As immediate results we get that (1) even the approximate version of Dempster’s rule of combination is hard to compute and (2) the hardness result for the Dempster-Shafer theory still holds even if we severely restrict the basic probability assignments allowed as evidence in the Dempster-Shafer theory.

2.4 Concluding Remarks

We have put results on the complexity of counting and approximating the number of satisfying assignments of propositional formulae in the context of various approximate reasoning techniques. The significance of this approach was illustrated by showing that various, supposedly different methods in approximate reasoning are equivalent to counting. It is worth noticing, for example, that while there is an active discussion in the approximate reasoning community as for differences in the semantical basis of the Dempster-Shafer theory and the Bayesian approach, (see, e.g., (Pearl, 1990; Shafer, 1990; Provan, 1992)) we show here that there is one computational problem underlying both approaches: computing inference is equivalent to counting satisfying assignments of a theory. Moreover, we have shown that this approach is valuable in evaluating techniques that use approximations in an effort to avoid computational difficulties. This was exemplified by analyzing heuristics used in constraint satisfaction problems and the utility of reasoning with approximate theories. We believe that the approach developed here can be found useful in the analysis of other problems of interest to AI.

Our hardness results indicate that for the problems of computing degrees of belief, probabilistic reasoning and other approximate reasoning techniques, one cannot expect to develop general-purpose algorithms that have a polynomial running time. Moreover, even computing approximate inference was proved to be intractable.

These results do not rule out the possibility of developing efficient algorithms that apply in restricted cases, as our positive results suggest; identifying more positive results and investigating how they apply to various techniques might be one direction to extend this work. We mention in particular the problem of (approximately) counting the number of satisfying assignments in bounded degree 2M0NCNF formulae. The problem is left open (see Figure 2.1) and its solution might be used to develop efficient algorithms for constraint satisfaction problems,

\(^{10}\)A similar result, using a different reduction, was proved independently by Provan (1990). We thank Greg Provan for bringing (Orponen, 1990; Provan, 1990) to our attention.
for example. The positive results presented here are important therefore not only for pointing out the tractability frontiers, but also since they provide a collection of techniques that can be used to further enhance our understanding of these problems and develop new results, possibly, for other problems of interest to AI.

However, the extent to which the hardness results apply (very restricted propositional languages, restricted topologies of Bayesian networks, etc.), imply that research should be directed away from investigating structural constraints on the “world” and towards investigating distributional constraints, or limiting our reasoning tasks rather than the “world” we reason about. The first might include constraining the distributions we can represent in our belief networks (e.g., (Poole, 1993)), while the second could imply studying restrictions on the type of queries we need to respond to. This is motivated also by the results in Section 2.3.3 that suggest that a possible approach to allow for efficient reasoning might be to constrain the queries (rather than the “world”). Indeed, partly motivated by these results, in Chapter 4 it is shown how constraining the queries side-steps the difficulties presented in Section 2.3.3 and leads to correct reasoning with approximate theories.

A possible interpretation of the surprising and widely applicable results presented here is that we need to reconsider the way we model the reasoning problem. One way to get around the difficulties presented here is to allow the reasoner other ways to access the “world”, instead, or in addition to the fixed (formula-based, Bayesian network-based, etc.) knowledge-based approach that we analyze here. This is the direction we pursue in the rest of this thesis, starting in the next chapter.

### 2.5 Proofs

In this section we formally state and prove the results outlined in Section 2.2. The results are stated only for some of the important languages in Figure 2.1 and results for other languages can be easily deduced by inclusion, as described in Section 2.2. The results in this section are summarized in three theorems: In Section 2.5.2 (Theorem 2.2.1) we prove results on the hardness of exact counting. In Section 2.5.3 (Theorem 2.2.2) we prove results on the hardness of approximate counting. In Section 2.5.4 (Theorem 2.2.3) we give positive results on exact counting by describing efficient algorithms for counting satisfying assignments for formulae in those languages.

#### 2.5.1 Preliminaries

For a Boolean function $f$ we denote by $M(f)$ the set of satisfying assignments of $f$ and by $|M(f)|$ its size. We denote $\{1, 2, \ldots , n\} = [n]$.  

29
Lemma 2.5.1 The problem of counting the number of satisfying assignments of a 2MONCNF is equivalent to the problem of counting the number of independent sets in a graph.

Proof: Given a graph $G(V,E)$ we associate with it a monotone CNF $\Phi_G$ on variables $\{x_1, \ldots, x_n\}$ as follows:

$$\Phi_G = \bigwedge_{(v_i,v_j) \in E} (x_i \lor x_j).$$

The inverse mapping is defined in the same manner: given a formula $\Phi \in 2\text{MONCNF}$ on $\{x_1, \ldots, x_n\}$, construct a graph $G_{\Phi}(V,E)$ on $\{v_1, \ldots, v_n\}$ as follows:

$$(v_i,v_j) \in E \iff (x_i \lor x_j) \text{ is a clause in } \Phi.$$ 

Now, let $S_I = \{v_j : j \in I\}$ be an independent set in $G$, then the assignment defined by

$$x_i = \begin{cases} 0 & \text{if } i \in I \\ 1 & \text{otherwise} \end{cases}$$

satisfies $\Phi_G$. The reason is that in every clause $x_i \lor x_j$ at least one of the variables is assigned to 1, since otherwise, by the definition of $\Phi_G$, $(v_i,v_j) \in E$, but both $v_i$ and $v_j$ are in the independent set $S_I$.

For the other direction, assume that $x$ is a satisfying assignment of $\Phi$, and let $I = \{i \in [n] : x_i = 0\}$, then $S_I = \{v_j : j \in I\}$ is an independent set in $G_{\Phi}$. The reason is that by the definition of the graph $G_{\Phi}$, no two vertices which share an edge are in $I$, since otherwise we have a clause in $\Phi$ that $x$ does not satisfy. \hfill \Box

Corollary 2.5.2 The problem of counting the number of satisfying assignments of a $k\mu$-2MONCNF is equivalent to the problem of counting the number of independent sets in a graph of degree $k$.

Proof: The mapping defined in Lemma 2.5.2 maps graphs of degree $k$ to $k\mu$ formulae. \hfill \Box

2.5.2 Exact Counting

Theorem 2.2.1: [Hardness of Exact Counting] Let $\Sigma \in \mathcal{L}$ be a propositional formula on $n$ variables. If $\mathcal{L}$ is one of the following propositional languages, counting the number of satisfying assignments of $\Sigma$ is complete for $\#P$:

1. $\mathcal{L} = 2\text{MONCNF}$
2. $\mathcal{L} = 2\text{BPMONCNF}$
3. $\mathcal{L} = 2\text{HORN}$
4. $\mathcal{L} = 3\mu-2\text{HORN}$
5. $\mathcal{L} = 4\mu-2\text{MON}$
Proof: (1) and (2) are well known: (1) is proved in (Valiant, 1979b); (2) is from (Provan and Ball, 1983). We can get (3) From (2), by negating all the variables in one of the bipartite sets.

To prove (4), given a formula \( \Phi \) in \( 2\text{HORN} \) we rewrite it, without changing the number of solutions, as a \( 3\mu\text{-2HORN} \) formula. Let \( \Phi \) be a \( 2\text{HORN} \) formula on \( \{x_1, \ldots, x_n\} \) and assume \( x_i \) appears \( m(i) \) times in \( \Phi \) (negated or unnegated). For every \( i \in [n] \) introduce \( m(i) \) new variables, \( \{x_i^{(j)}\}_{j=1}^{m(i)} \) and replace the \( j \)th appearance of \( x_i \) in \( \Phi \) by \( x_i^{(j)} \) to get \( \Sigma_i \) a \( \mu\text{-2HORN} \) formula. We then conjoin \( \Sigma \) with \( \Lambda_i \Gamma_i \), where \( \Gamma_i \) is the following \( 2\mu\text{-2HORN} \) formula:

\[
\Gamma_i = \overline{x_i^{(1)}} \rightarrow x_i^{(2)} \rightarrow \cdots \rightarrow x_i^{(m-1)} \rightarrow \overline{x_i^{(m)}} \rightarrow x_i^{(1)}.
\]

Clearly

\[
\Phi \equiv \Sigma \land \bigwedge_i \Gamma_i.
\]

Thus, the number of satisfying assignments of the \( 3\mu\text{-2HORN} \) formula \( \Sigma \land \bigwedge_i \Gamma_i \) is equal to the number of satisfying assignments of the original \( 2\text{HORN} \) formula, and the counting problems for those languages are therefore equivalent.

For (5) we use a different rewriting technique. Given a \( 3\mu\text{-2CNF} \) formula \( \Phi \) we rewrite it, while preserving the number of satisfying assignments as \( \Sigma \land \neg\Gamma \) where \( \Sigma \) and \( \Gamma \) are both monotone, and \( \Sigma \land \Gamma \) are both in \( 4\mu\text{-2MON} \). Since \( |\Sigma \land \neg\Gamma| = |\Sigma| - |\Sigma \land \Gamma| \), the hardness of exact counting for \( 4\mu\text{-2MON} \) formulae results from the hardness of counting for \( 3\mu\text{-2CNF} \) formulae (cf. (4)).

In rewriting \( \Phi \), given a variable \( x_i \), which appears both negated and unnegated in \( \Phi \), we replace its (up to 2) unnegated occurrences by \( y_i \) and its (up to 2) negated occurrences by \( z_i \). The resulting formula is a \( 3\mu\text{-2MONCNF} \) formula \( \Phi' \). To force that \( \forall i, \overline{y_i} = z_i \) we denote

\[
\Phi'' = \bigwedge_i (y_i \lor z_i) \quad \Gamma = \bigvee_i (y_i \land z_i).
\]

It is clear that

\[
\Phi \equiv \Phi' \land \Phi'' \land \neg\Gamma = \Sigma \land \neg\Gamma.
\]

Since \( \Sigma = \Phi \land \Phi'' \) is a \( 3\mu\text{-2MONCNF} \) formula and \( \Gamma \) is a \( 1\mu\text{-2MON} \) formula (in a DNF form), the result follows.

2.5.3 Approximate Counting

**Theorem 2.2.2:** [Hardness of Approximation] Let \( \Sigma \in \mathcal{L} \) be a propositional formula on \( n \) variables. If \( \mathcal{L} \) is one of the following propositional languages, approximating the number of satisfying assignments of \( \Sigma \) to within a factor of \( 2^{n^{1-\epsilon}} \), for any fixed \( \epsilon > 0 \), is NP-hard.

(1) \( \mathcal{L} = 2\text{MONCNF} \)

(2) \( \mathcal{L} = 3\mu\text{-2HORN} \)
**Proof:** To get (2) from (1) we use the rewriting technique for non-monotone clauses as in (4) of Theorem 2.2.1. (This technique leaves the number of solutions the same but might increase the number of variables up to $n^2$. This can easily be handled as we do below). Notice that the rewriting technique used in (5) of that theorem does not extend for approximations.

The next lemma provides the main step in the proof of (1). The proof is based on the “blow-up” technique developed in (Jerrum, Valiant, and Vazirani, 1986). The lemma is a variant of one that appears in (Sinclair, 1988).

**Lemma 2.5.3** For any $\epsilon$, approximating the number of independent sets of a graph on $n$ vertices within $2^{n^{1-\epsilon}}$ is NP-hard.

**Proof:** We use the “blow-up” technique introduced in (Jerrum, Valiant, and Vazirani, 1986), to reduce the problem of approximating the number of independent sets in $G$ to the $k$-INDEPENDENT-SET problem (Garey and Johnson, 1979). Given $G(V, E)$, where $|V| = n$, we construct a graph $G'(V', E')$ such that approximating the number independent sets in $G'$ to within $2^{n^{1-\epsilon}}$ can be used to solve $k$-INDEPENDENT-SET in $G$. $G'$ is defined as follows: each vertex $v \in V$ is blown-up to a “cloud” $c(v)$ of $m$ vertices in $G'$. If $(u, v) \in E$, in $G'$ we construct a complete bipartite graph on $c(v) \cup c(u)$; otherwise, there are no edges connecting $c(v)$ to $c(u)$.

Formally,

$$V' = \{v^j : v \in V; j \in \{1, \ldots, m\}\}$$

$$E' = \{(v^i, u^j) : (v, u) \in E; i, j \in \{1, \ldots, m\}\}$$

Assume now that $G$ contains an independent set $I$ of size $k$. Then, $I' = \{v^j : v \in I; j \in \{1, \ldots, m\}\}$ is an independent set of size $km$ in $G'$. Since all the subsets of an independent set are also independent sets, there are at least $N_{\min} = 2^{km}$ independent sets in $G'$.

On the other hand, if $G$ contains no independent set of size $k$, an independent set in $G'$, contains vertices from up to $k-1$ “clouds” since otherwise, the corresponding vertices in $G$ (the “projection” of the clouds) generate in $G$ an independent set of size larger than $k-1$. In particular, the largest independent set in $G'$ is of size $\leq (k-1)m$ (there might be, however, many different independent sets of that size). Thus, there are no more than $N_{\max} = \left(\frac{n}{k-1}\right)^2 2^{(k-1)m}$ independent sets in $G'$.

Finally, let $k = n/2$; in this case, $k$-INDEPENDENT-SET is NP-hard ((Garey and Johnson, 1979), p. 194). Given $\epsilon > 0$, choose $r$ large enough such that $1 - \frac{r^2}{r+1} < \epsilon$, and let $m = n^r$. The “blow-up” graph $G'$ is of size $|V'| = nm = n^{r+1}$. We have that,

$$\sqrt{N_{\min}/N_{\max}} = \sqrt{\frac{2^{km}}{\left(\frac{n}{k-1}\right)^2 2^{(k-1)m}}} = \sqrt{\frac{2^m}{\left(\frac{n}{k-1}\right)^2}} = \sqrt{\frac{2^{2r}}{2^{n/2}}} = \sqrt{\frac{2^{2r}/4n}{2^{n/2}}} = 2^{r^2-n} = 2^{n^2/4} n^{r^2/2}.$$  

Notice also that this “blow-up” procedure is polynomial in the size of the original graph. Therefore, if we can approximate the number of independent sets in $G'$ within
we can use this approximation to decide whether the graph $G'$ has more than $N_{\text{min}}$ or less than $N_{\text{max}}$ independent sets. As argued above, this leads to deciding $n/2$-INDEPENDENT SET.

We have proved in Lemma 2.5.1 that counting the number of satisfying assignments of a 2MONCNF is equivalent to the problem of counting the number of independent sets in a graph. This, together with Lemma 2.5.3 implies the theorem.

2.5.4 Positive Results

Theorem 2.2.3: [Positive Results] Let $\Sigma \in \mathcal{L}$ be a propositional formula on $n$ variables. If $\mathcal{L}$ is one of the following propositional languages, there exists an efficient algorithm for counting the number of satisfying assignments of $\Sigma$.

1. $\mathcal{L} = 2\mu$-2MONCNF
2. $\mathcal{L} = 2\mu$-2CNF
3. $\mathcal{L} = \text{Acyclic-2MONCNF}$
4. $\mathcal{L} = \text{Acyclic-2HORN}$

Proof: We prove (1) by developing a closed form formula that is easy to evaluate for the number of independent sets in graphs of degree 2. For the other cases we develop efficient algorithms. (3) is the problem of counting independent sets of trees, for which we give an efficient recursive algorithm. The algorithms for (2) and (4) are more elaborate. In both cases we start by constructing chains of the form $x_1 \rightarrow x_2 \rightarrow \ldots \rightarrow x_r$, from the original theory. We then show that the original theory can be represented as a composition of these chains, and develop compositions rules that allow us to count the number of satisfying assignments of the composite chains. The difference between (2) and (4) is the type of compositions allowed. We note that (1) and (3) are sub-cases of (2) and (4), respectively, but we give for them separate proofs, since those cases are considerably simpler.

Proof of (1):

Based on Corollary 2.5.2 it is enough to count the number of independent sets of a degree 2 graph. In the following we consider the empty set to be an independent set.

Let $G$ be a connected graph of maximal degree 2 on $n$ vertices. $G$ can be either a cycle, in case all its vertices are of degree 2, or an arm if exactly 2 of its vertices are of degree 1. We have:

Lemma 2.5.4 Let $IS_n^c$ denote the number of independent sets in a cycle of length $n$, and $IS_n^a$ the number of independent sets in an arm of length $n$. Then,

(i) $IS_n^a = 1 + \sum_{j=1}^{\lfloor n/2 \rfloor} \binom{n-j+1}{j}$
(ii) $IS_n^c = 1 + \sum_{j=1}^{\lfloor n/2 \rfloor} \binom{n-j}{j} + \binom{n-j-1}{j-1}$
Proof: We denote by $IS_{n,j}^a$ (respectively, $IS_{n,j}^c$) the number of independent sets of size $j$ in an arm (respectively, cycle) of length $n$. $IS_{n,j}^a(v)$ counts those independent sets that contain a fixed vertex, $v$, in the cycle.

(i) The problem of computing $IS_{n,j}^a$ reduces to the following combinatorial problem: find the number of selections of $j$ integers from the set $\{1, \ldots, n\}$, such that no two consecutive numbers are selected. To count this number, consider any selection of $j$ different numbers from among $\{1, \ldots, n-j+1\}$. The mapping which adds 0 to the first number selected, 1 to the second, $\ldots$, $j-1$ to the $j$th, is a 1-1 correspondence between those selections and the legal selections we count. Thus, we get:

$$IS_{n,j}^a = \binom{n-j+1}{j}.$$ 

To get the total number we sum on $j$, $j \leq \lfloor n/2 \rfloor$, and add 1 for the empty set.

(ii) We claim that $IS_{n,j}^c = IS_{n-1,j}^a + IS_{n-1,j}^a(v)$. To see that, consider the cycle as an arm with end points $v_1, v_n$. (i.e., $v_1, v_n$ are adjacent in the cycle). The first term corresponds to the independent sets of the cycle that do not contain, say, $v_n$, while the second term correspond to those that contain it. (The latter group cannot contain $v_1$, so we get them by shifting each set that contains $v_1$ by one place). Also, $IS_{n-1,j}^a(v) = IS_{n-j-1,j-1}^a$ as we can just add two adjacent vertices as prefix, one that is not selected and $v$. We get:

$$IS_{n,j}^c = \binom{n-j}{j} + \binom{n-j-1}{j-1}.$$ 

To get the total number we sum on $j$, $j \leq \lfloor n/2 \rfloor$, and add 1 for the empty set. \hfill \Box

As an immediate consequence of Lemma 2.5.4 we get:

**Lemma 2.5.5** Let $G(V, E)$ be a graph of maximal degree 2, and assume $G$ has $r$ connected components, of sizes $n_1, \ldots, n_r$, respectively. The number of independent sets in $G$ is

$$\prod_{i=1}^r IS_{n_i}^x,$$

where $x \in \{a, c\}$ depends on whether the component is an arm or a cycle.

**Proof of (2):**

All the clauses in a 2μ-2CNF theory are of the form $l_1 \rightarrow l_2$ where every literal $l_i$ might be a variable $x \in X_n$ or its negation, and every variable appears no more than twice in the given theory. (I.e., either a literal appears twice and its negation never appears in the theory or that the literal and its negation appear once each.) Notice that in the implication representations every clause $\overline{l_1} \lor l_2$ has two equivalent representation, $l_1 \rightarrow \overline{l_2}$ and $\overline{l_2} \rightarrow \overline{l_1}$. Since it will be convenient to use the implication representation we assume that we hold both representations and use the one that it
more convenient. If a literal \( l \) appears both as an antecedent and as a consequent in two clauses, e.g., \( l_1 \rightarrow l_2 \) and \( l_2 \rightarrow l_3 \) we can combine then the chain \( l_1 \rightarrow l_2 \rightarrow l_3 \), which now contains the only occurrence of \( \text{var}(l_2) \) in the theory.

We call the theory \( C = l_1 \rightarrow l_2 \rightarrow \ldots \rightarrow l_r \) a simple chain. In this case, \( l_1 \) is the antecedent of \( C \) and \( l_r \) its consequent. The antecedent and the consequent are the only literals of degree 1 in the theory; all other literals are of degree 2. The antecedent and the consequent literals might have the same underlying variable, but this cannot be the case for other literals in the chain. We say that a simple chain \( l_1 \rightarrow l_2 \rightarrow \ldots \rightarrow l_r \) in a 2\( \mu \)-2CNF theory is maximal if it cannot be extended, that is, the theory contains no other clause with \( l_1 \) as consequent and no other clause with \( l_r \) as antecedent. Notice that two maximal simple chains \( C_1 \) and \( C_2 \) in a theory cannot have as consequents \( l_1 \) and \( \overline{l_1} \), respectively. The reason is that in this case we can “reverse” \( C_2 \) and negate all its literals, to get an equivalent chain that has \( l_1 \) as antecedent. The new chain can be concatenated to \( C_1 \), contradicting its maximality. A simple chain is called a cycle if \( \text{var}(x_1) = \text{var}(x_r) \); in this case all variables are of degree 2.

Let \( C_1, C_2, \ldots, C_k \) be maximal simple chains in a 2\( \mu \)-2CNF theory \( \Sigma \). Assume \( C_1 \) and \( C_2 \) both have \( l_1 \) as antecedent. In this case we can compose the simple chains, and say that \( C_1 \wedge C_2 \) is a composite chain. Similarly, \( C_1 \) and \( C_2 \) can be composed if they have a common consequent. Notice that if a literal \( l \) appears in two maximal simple chains, it must appear in it both as an antecedent or as a consequent. (Being internal to both contradicts the degree requirement while being a consequent in one and an antecedent in the other contradicts the maximality of the chain.) Thus, we can repeat this process of composition until there are no two chains in \( \Sigma \), simple or composite, that share a variable.

If two chains share both antecedents and consequents, composing them results in a closed composite chain. Every composite chain \( C \) that is not closed has exactly two literals of degree 1, say \( l_i \) and \( l_j \). We decide arbitrarily to denote \( l_i = t(C) \), the tail of \( C \) and \( l_j = h(C) \), the head of \( C \) if \( i < j \). The tail and head of a composite chain can both antecedents, both consequents or any other combination.

Given a chain \( C_1 \) on variables \( \{x_1, x_2, \ldots, x_k\} \) (i.e., all \( k \) variables appear in \( C_1 \)), denote by \( N_{C_1} \) the number of assignments of \( \{x_1, x_2, \ldots, x_k\} \) that satisfy \( C_1 \). Likewise, for \( b \in \{0, 1\} \) denote by \( N_{C_1|t=b} \), \( N_{C_1|b=b} \) the number of assignments of \( \{x_1, x_2, \ldots, x_k\} \) that satisfy \( C_1 \), given that we force the head (respectively, tail) literal to 0 or 1, and by \( N_{C_1|t=b_1, h=b_1} \) when we force both head and tail to some value in \( \{0, 1\} \).

Notice that given the values \( N_C \), \( N_{C|t=1} \), \( (N_{C|b=1}) \) and \( N_{C|t=1, h=1} \) one can determine all the possible values \( N_{C|t=a, h=a} \). Therefore, it will be necessary to compute all these values for a composite chain. For a closed chain, it will be enough to compute \( N_C \), since the chain will not be composed any more.

Given a 2\( \mu \)-2CNF theory, to count the number of its satisfying assignments we first decompose it to simple chains and cycles. The number of satisfying assignments of these simple theories is given in Lemma 2.5.7. As argued above, if a variable \( x \)
is common to two chains it must be an antecedent in both or a consequent in both. We derive, in Lemma 2.5.8 and Lemma 2.5.9 a composition rule that shows how to compute the number of satisfying assignments of the conjunction of two theories, under the restriction that these two theories are part of a $2\mu$-2CNF theory. This composition rule is applied also for conjuncting composite chains, until there are no more compositions to be made. At this point the theory is represented as a conjunction of disjoint theories, and we use Lemma 2.5.6 to compute its total number of satisfying assignments. We now describe the algorithm in some more details, and then prove some lemmas that show its correctness.

**Algorithm: Count $2\mu$-2HORN**

Let

$$\Sigma = (l_{i_1} \lor l_{j_1}) \land \ldots \land (l_{i_m} \lor l_{j_m})$$

be CNF theory in $2\mu$-2CNF such that $\text{var}(l_{i_k}), \text{var}(l_{j_k}) \in X_n$. The following procedure counts the number of satisfying assignments of $\Sigma$:

**Construct simple chains:**

- Represent each clause\(^{11}\) as

  $$l_1 \lor l_2 \equiv \overline{l}_1 \rightarrow l_2 \equiv \overline{l}_2 \rightarrow l_1$$

- Fix an order of the clauses. Start from the first clause and greedily combine $l_i \rightarrow l_j$ and $l_j \rightarrow l_k$ to $l_i \rightarrow l_j \rightarrow l_k$. (That is, for each clause, check if one of its representations can be combined in that way, if any). Go on until you end up with a maximal simple chain.

- Starting from the next available clause, repeat the above procedure, using only clauses that are not already part of previously constructed chains. Go on until no more combination can be made. (I.e., no variable occurs both as a consequent and as an antecedent.) Make sure in this process that the constants $T$ and $F$ are never internal to a chain.

- For each simple chain of the form

  $$C = l_{i_1} \rightarrow l_{i_2} \rightarrow \ldots \rightarrow l_{i_r}$$

  compute, using Lemma 2.5.7 the value of $N_C, N_{C|l_i=1}, N_{C|l_i=1}, N_{C|l_i=1, l_j=1}$.

Notice that in the above process no more than $n$ combination steps are required (since the degree of a variable in $\Sigma$ is at most 2), and that the resulting chains are uniquely defined.

\(^{11}\)Clauses of length 1 can also be represented in this way, e.g., $x \equiv (T \rightarrow x)$ and $\overline{x} \equiv (x \rightarrow F)$. The fact that the constants $T$ and $F$ might appear in the theory more than twice will not affect the correctness of the algorithm. (See remark on that later.)
Combine chains:

- Given $\Sigma$, represented as a conjunction of simple chains, combine chains $C_1$ and $C_2$ if they have a common variable as antecedent, as consequent, or both.
- At each combination step, resulting in a composite chain $C$, compute, using Lemma 2.5.8 and Lemma 2.5.9, the values of $N_C, N_C|_{t=1}, N_C|_{h=1}, N_C|_{t=1, h=1}$.
- Go on until there are no chains with common variables. No more than $n$ combinations steps are required, since the degree of a variable in $\Sigma$ is at most 2. The process results in disjoint composite chains.

Compute the number of satisfying assignments: Let $\{C_1, C_2, \ldots, C_k\}$ be the set of disjoint composite chains given by the previous stage. Let $N_{C_i}$ be the number of assignments that satisfy $C_i$, counted only over the variables in $C_i$. If only $r$ variables from $X_n$ are used in $\{C_1, C_2, \ldots, C_k\}$ Then, using Lemma 2.5.6, the number of satisfying assignments of $\Sigma$ is

$$|M(\Sigma)| = 2^{n-r} \prod_{j=1}^{k} N_{C_j}$$

Correctness

It is clear that the maximal simple chains constructed by the algorithm are unique, and so are the composite chains. It is also clear that the construction is efficient. We just need to show how to derive the number of satisfying assignments in that process. We show that in the next lemmas.

**Lemma 2.5.6** Let $\Sigma = C_1 \land C_2 \land \ldots \land C_k$ be a formula on $X_n$ such that for each $i$, $x_i$ appears in exactly one of the conjuncts $C_j$. Let $N_{C_i}$ be the number of independent sets of $C_i$ and $N_\Sigma$ the number of independent sets of $\Sigma$. Then,

$$N_\Sigma = \Pi_{i=1}^{k} N_{C_i}$$

**Proof:** Clear from the fact that $C_i$ are variable-disjoint. $\diamond$

The initial step in computing the number of satisfying assignments is given by the next lemma:

**Lemma 2.5.7** Consider the simple chain

$$C = l_1 \rightarrow l_2 \rightarrow \ldots \rightarrow l_r$$

(i) If all the underlying variables in $C$ are different then $N_C = r + 1$, $N_C|_{t=1} = 1$, $N_C|_{h=1} = r$, $N_C|_{t=1, h=1} = 1$.
(ii) If $l_1 = l$, then $N_C = 2$.
(iii) If $l_1 = \overline{l}$, then $N_C = 1$.
Proof: For (i) we note that if \( l_i = 1 \) then \( l_j = 1 \) for \( j > i \), so we need to consider only the first index \( i \) such that \( l_i = 1 \). There are \( r \) possibilities for that and one satisfying assignment in which all variables are 0. The other statements follow similarly (here \( l_i = t(C) \); \( l_r = h(C) \)). In (ii), either all literals are 1 or all are 0. In (iii) we must have that: \( l_1 = l_2 = \ldots = l_{r-1} = 0 \). \( \Box \)

The next lemma shows how to compute the number of satisfying assignments when composing two chains.

Lemma 2.5.8 Let \( C_1, C_2 \) be two composite chains that have a exactly on variable, \( x \), in common. We assume, without loss of generality, that the common variable is the tail of both \( C_1 \) and \( C_2 \), that the tail of the composite chain \( C_1 \land C_2 \) is the degree-1 variable coming from \( C_1 \) and the head is the degree-1 coming from \( C_2 \).

(i) If \( x \) appears as a positive (negative, respectively) variable in both chains it must be either an antecedent in both chains or a consequent in both. In this case,

\[
- N_{C_1 \land C_2} = N_{C_1 | t=1} N_{C_2 | t=1} + N_{C_1 | t=0} N_{C_2 | t=0}
\]

(ii) Assume \( x \) appears as a positive variable in \( C_1 \) and as a negative variable in \( C_2 \). In this case \( x \) must be an antecedent in one of the chains (say, \( C_1 \)), and consequent in the other. We have:

\[
- N_{C_1 \land C_2} = N_{C_1 | t=1} N_{C_2 | t=0} + N_{C_1 | t=0} N_{C_2 | t=1}
\]

Proof: The proof is immediate from the notational assumption made and the observation that all the possible satisfying assignments are counted that way, and no other satisfying assignment is possible. \( \Box \)

Lemma 2.5.9 Let \( C_1, C_2 \) be two composite chains that have exactly two variables, \( x, y \), in common. We assume, without loss of generality, that the literals whose variable is \( x \) are in the tail of both \( C_1 \) and \( C_2 \), and those whose variable is \( y \) are in the head of both chains. Since the result of this composition is a closed chain it is sufficient to compute \( N_{C_1 \land C_2} \).
(i) If both $x$ and $y$ appear as a positive (negative, respectively) variables in both chains (i.e., each must be either an antecedent in both chains or a consequent in both) we have:

$$N_{C_1 \land C_2} = \sum_{b_1, b_2 \in \{0, 1\}} N_{C_1 | b_1 = b_2, h = b_2} N_{C_2 | b_1 = b_1, h = b_2}$$

(ii) If both occurrences of $x$ are positive and $y$ appears once as a positive variable and once negated we have:

$$N_{C_1 \land C_2} = \sum_{b_1, b_2 \in \{0, 1\}} N_{C_1 | b_1 = b_2, h = b_2} N_{C_2 | b_1 = b_1, h = \overline{b_2}}$$

(iii) If both $x$ and $y$ appear as positive variable in one chain and negated in the other, then we have:

$$N_{C_1 \land C_2} = \sum_{b_1, b_2 \in \{0, 1\}} N_{C_1 | b_1 = b_2, h = b_2} N_{C_2 | b_1 = b_1, h = \overline{b_2}}$$

As an example, consider the case of composing two simple maximal chains

$$C_1 \land C_2 = (x_1 \rightarrow x_2 \rightarrow \ldots \rightarrow x_{r_1 - 1} \rightarrow z) \land (y_1 \rightarrow y_2 \rightarrow \ldots \rightarrow y_{r_2 - 1} \rightarrow z),$$

where the $x_j$'s are different from the $y_j$'s, $z$ is the tail variable in both chains, and $x_1, y_1$ are the tail and the head, respectively of the composite chain. It is easy to see that

$$N_{C_1 \land C_2} = r_1 r_2 + 1$$

$$N_{C_1 \land C_2 | b_1 = 1} = N_{C_1 \land C_2 | b_1 = 1} = r_2$$

$$N_{C_1 \land C_2 | b_1 = 1} = N_{C_1 \land C_2 | b_1 = 1} = r_1$$

$$N_{C_1 \land C_2 | b_1, b_2 = 1} = 1.$$

If also $x_1 = y_1$ then $N_{C_1 \land C_2} = (r_1 - 1)(r_2 - 1) + 2.$

**Proof:** The proof is immediate from the notational assumption made and the observation that all the possible satisfying assignments are counted that way, and no other satisfying assignment is possible. \(\diamondsuit\)

With the observations that the computation above can be done in time polynomial in the size of the formula, this completes the correctness proof for algorithm Count-2\(\mu\)-2CNF. We note that in the case of clauses that contain a constant, $T$ or $F$, since all of them appear either as tail or head of a chain the algorithm can handle multiple occurrences of them. This in fact is true in general. The composition rule holds if we require only that the degree of variables that appear internal to a simple chain is at most 2, while the degree of all other variables, those whose all occurrences in the theory is either positive or negative, is not restricted. The
Proof of (3):

By Lemma 2.5.1, given $\Sigma \in \text{Acyclic-2MONCNF}$, it is sufficient to count the number of independent sets in the corresponding graph which is, by the definition of an Acyclic-2MONCNF formula (Section 2.2), an acyclic graph. We first consider the case of a connected acyclic graph, a tree.

**Lemma 2.5.10** Let $T$ be a tree on $n$ vertices. The number of independent sets of $T$ can be computed in time $O(n)$.

**Proof:** Let $T$ be a tree with root $r$. For a vertex $x \in T$, we denote by $T_x$ the subtree of $T$ with $x$ as root. $c(x)$ denotes the set of all vertices which are children of $x$ in $T$, and $gc(x)$ the grandchildren of $x$ in $T$. We denote by $IS_x$ the number of independent sets of the tree rooted at $x$. Among these independent sets, $IS_x(x)$ denotes the number of those which contain the root $x$, and $IS_x(\not x)$ denotes the number of those which do not contain the root $x$.

Notice, that for all $y \in c(x)$ and any independent set $I$ of $T_x$, $I_y = \{z \in I \cap T_y\}$ is an independent set in $T_y$. We use this in the next claim to represent the number of independent sets in $T_x$ in terms of the number of independent sets of subtrees of $T_x$.

**Claim 2.5.11** For $IS_x, IS_x(x), IS_x(\not x)$ as defined above, we have:

(i) $IS_x(\not x) = \prod_{y \in c(x)} IS_y$

(ii) $IS_x(x) = \prod_{y \in gc(x)} IS_y$

(iii) $IS_x = IS_x(x) + IS_x(\not x) = \prod_{y \in gc(x)} IS_y + \prod_{y \in c(x)} IS_y$.

**Proof:** If $c(x) = \{y_1, y_2, \ldots, y_k\}$ and $\{I_y \subseteq T_y\}_{i=1}^k$ any collection of independent sets (in the respective trees) then $\bigcup_{y \in c(x)} I_y$ is an independent set in $T_x$ that does not contain $x$. For the other direction, clearly any independent set $I \subseteq T_x$ that does not contain $x$ can be decomposed uniquely as above. For (ii), similarly, if $I$ is an independent set in $T_x$ and $x \in I$, then clearly $\forall z \in gc(x), I_z = \{z \in I \cap T_z\}$ is an independent set in $T_z$. For the other direction, if $gc(x) = \{z_1, z_2, \ldots, z_k\}$ and $\{I_z \subseteq T_z\}_{i=1}^k$ any collection of independent sets in $T_z$, then clearly $\bigcup_{z \in gc(x)} I_z \cup \{x\}$ is an independent set in $T_x$, since it contains no vertex from $c(x)$. (iii) is immediate from (i) and (ii).

We now present an algorithm, **Count-IS-Tree**, that computes the number of independent sets of a given tree. We denote by $r(x)$ the rank of the vertex $x \in T$. The

---

12As before, we count the empty set as one of the independent sets.
rank of $x$ is defined as follows: If $x$ is a leaf, $r(x) = 0$. If $x$ is an internal node in the tree we define, $r(x) = 1 + \max_{y \in c(x)} r(y)$. We denote by $R_i$ the set of all vertices $x \in T$ such that $r(x) = i$, and assume w.l.o.g. that the tree $T$ is represented as a collection of its sets $R_i$. In the algorithm, we compute the number $IS_x$ of independent sets of a tree rooted at $x \in R_i$, given the values $IS_y$ for all $y \in R_j$, for $j < i$.

**Count-IS-Tree($T$):**

For all $x \in R_0$, $IS_x = 2$.
For all $x \in R_1$, $IS_x = 1 + 2^{l(x)}$.
For $i = 3, 4, \ldots$, do:
   For all $x \in R_i$, $IS_x = \prod_{z \in c(x)} IS_z + \prod_{y \in c(x)} IS_y$
End

Figure 2.3: Counting independent sets of a tree

For the boundary conditions, notice that if the tree contains a single vertex $x$, then $IS_x = 2$, and if the children of the root are leaves, then $IS_x = 1 + 2^\# \text{leaves}$. Thus, the correctness of the algorithm follows from Claim 2.5.11 and the discussion above, and this completes the proof of the lemma. \hfill \diamondsuit

Since an acyclic undirected graph is a union of disjoint trees, using Lemma 2.5.6 completes the proof.

**Proof of (4):**

As in the proof of (3) we assume that in the graph that corresponds to the 2HORN formula $\Sigma$ every connected component is a tree. The counting algorithm is very similar to the one presented for the Acyclic monotone case. We prove a claim that is analog to Claim 2.5.11, and use it to count the satisfying assignments as in the algorithm **Count-IS-Tree($T$)** above.

**Claim 2.5.12** Let $N_x$ denote the number of assignments that satisfy the conjunction of clauses that correspond to a tree rooted at $x$ (with respect to these variables only). $N_x(0)$ (respectively, $N_x(1)$) denotes those assignments in which $x$ is assigned $0$ (respectively, $1$). We have that:

(i) $N_x(0) = \prod_{y \in c(x)} N_y$
(ii) $N_x(1) = \prod_{y \in c(x)} N_y(1)$
(iii) $N_x = N_x(0) + N_x(1) = \prod_{y \in c(x)} N_y + \prod_{y \in c(x)} N_y(1)$

**Proof:** To prove (i) we observe that since $x$ is assigned 0, there are no restrictions on the satisfying assignments of the tree rooted at $y \in c(x)$. Since the subtrees rooted at different elements of $c(x)$ are disjoint, we get the result. We get (ii) by observing that an assignment satisfies the formula corresponding to $T_x$, where $x$ is assigned 1, iff all $y \in c(x)$ are assigned 1. (iii) is immediate from (i) and (ii). \hfill \diamondsuit
Noticing that if the corresponding tree is of depth 1, the number of satisfying assignments is $1 + 2^{\# \text{ of leaves}}$, serves as the boundary condition for the procedure, that uses Claim 2.5.12 to count the number of satisfying assignments of the Acyclic-2HORN formula $\Sigma$. Similar to the algorithm $\text{Count-IS-Tree}(T)$ in the proof of (3) above we get an algorithm that computes the number of satisfying assignments efficiently.
There must be other shifts.
Otherwise it would be quite hopeless.

Samuel Beckett, The Unnamable

Consider a baby robot, starting out its life. If it were a human being, nature would have provided for the infant a safe environment in which to spend an initial time period. In this period she adapts to her environment and learns about the structures, rules, meta-rules, superstitions and other information the environment provides. In the meantime, the environment protects her from fatal events. Only after this “grace period” is she expected to have “full functionality” in her environment, and her performance will depend on the world she grew up in and reflects the amount of interaction she has had with it.

Computational learning theory, a subfield concerned with modeling and understanding learning phenomena (Valiant, 1984), takes a similar view that the performance of the learner should be evaluated relative to the world, but only after a certain learning period. Early theories of intelligent systems, however, had assumed that cognition (namely, computational processes like reasoning, language recognition, object identification and other “higher level” cognitive tasks) can be studied separately from learning (See (Kirsh, 1991) for a discussion of this issue.).

Reasoning in intelligent systems has been studied mostly in the knowledge-based system framework (McCarthy, 1958). There, it is assumed that the knowledge is given to the system, stored in some representation language with a well defined meaning, and that there is a reasoning mechanism, that can be used to determine what can be inferred from the sentences in the $KB$. The question of how this knowledge might be acquired and whether this should influence how the performance of the reasoning system is measured is not considered. The intuition behind this approach is based on the following observation:
**Observation:** If there is a learning procedure that can learn an exact description of the world in representation \( R \), and there is a procedure that can reason exactly using \( R \), then there is a complete system that can learn to produce “intelligent behavior” using \( R \).

We believe that the separate study of learning and the rest of cognition is, at least partly, motivated by the false assumption that the converse of the above observation also holds, namely, that if there is a system that can learn to produce “intelligent behavior”, then there is a learning procedure that can learn a representation of the world, and a reasoning procedure that can reason with it.

Computational considerations, however, render this self-contained reasoning approach as well as other variants of it inadequate for common-sense reasoning. As was shown in the previous chapter this is true not only for the task of deduction, but also for many other forms of reasoning which have been developed, partly in order to avoid the computational difficulties in exact deduction and partly to meet some (psychological and other) plausibility requirements. All those were shown to be even harder to compute than the original formulation. The same is true for a lot of recent work in reasoning that aims at identifying classes of limited expressiveness, with which one can perform some sort of reasoning efficiently (Brachman and Levesque, 1984; Levesque, 1992; Selman, 1990). None of these works meet the strong tractability requirements for common-sense reasoning as described, for example, in (Shastri, 1993), even though, (as argued, for example, in (Doyle and Patil, 1991)) the inference deals with limited expressiveness and is sometimes restricted in implausible ways.

An additional motivation for this work is the current disconnected state of the fields of learning and reasoning. Perhaps the most important open question in learning theory today is that of learning DNF or CNF formulas (these problems being equivalent). However, even if one had a positive learnability result for these classes, this would be relevant only for classification tasks, and could not be used for reasoning. The reason is that if the output of the learning algorithm is a CNF expression, then reasoning from an expression of this form is computationally hard. From a traditional reasoning point of view, on the other hand, learning a DNF is not considered interesting, since it does not relate easily to a rule based representation. In this work, therefore, while we build on the framework and some of the results of computational learning theory, we distinguish this, now traditional, learning task which we call here Learning to Classify from the new learning task, Learning to Reason.

**A Motivating Example:**
Consider the following example (this will be presented more rigorously later in the chapter): Let \( S = \{0, 1\}^n \) be an instance space and let \( D \) be some probability distribution defined on it. The conditional probability \( Pr_D(\alpha|KB) \) is the degree of belief in the query \( \alpha \), given the knowledge base \( KB \). Consider the problem
of computing the conditional probability $Pr_D(\alpha|KB)$ where KB and $\alpha$ are some propositional CNF formulas. For very restricted cases of knowledge bases $KB$, and for the case where $D$ is the uniform distribution we have shown in the previous chapter that:

**Theorem:** The problem of computing $Pr_D(\alpha|KB)$ is $\#P$-Complete. Approximating it is NP-hard.

Consider the following approach to the problem of computing $Pr_D(\alpha|KB)$. Assume the existence of an Example Oracle $EX_D$ that when accessed returns a sample $x \in S$, taken randomly according to the distribution $D$.

**Algorithm Estimate:**

- Use $EX_D$ to take $m$ samples from $S$.
- Let $\hat{Pr}_D(KB)$ be the number of samples taken that satisfy $KB$.
- Let $\hat{Pr}_D(\alpha \land KB)$ be the number of samples taken that satisfy $\alpha \land KB$.
- Let $\hat{Pr}_D(\alpha|KB) = \frac{\hat{Pr}_D(\alpha \land KB)}{\hat{Pr}_D(KB)}$

We denote by $D(KB)$ the measure of the set of satisfying assignments of $KB$ under $D$ and by $Q$ the class of propositional languages from which the query $\alpha$ is taken. Using standard learning theory arguments it is easy to show:

**Theorem:** Let $0 < \epsilon, \delta < 1$ be given and assume that the number of samples taken is $m = \frac{1}{\epsilon} (\ln |Q| + \ln \frac{1}{\delta})$. If $D(KB) > \frac{1}{p(n)}$ for some polynomial $p(n)$, then with probability $> 1 - \delta$, the estimate $\hat{Pr}_D(\alpha|KB)$ approximates the true probability $Pr_D(\alpha|KB)$ within ratio of $p(n) \epsilon$. In particular, the number of samples required to achieve this approximation is polynomial, and therefore the approximation algorithm is polynomial.

(In Section 3.2 we state this result again in a slightly different way, making the argument a little more formal and prove it.)

Intuitively, the result means that if we are exposed to enough examples from some context (governed by the context distribution $D$), we can perform correct and efficient reasoning within this context.

Thus, this example reveals that having direct access to the “world”, rather than to some abstraction of the “world” information into some (logical, or other) knowledge representation, is powerful. It avoids the problem of finding models of the “world”, which is the main source of computational difficulty in the current formulation of the reasoning problem, and by that supports efficient reasoning.

**Our Approach:**

We argue that the main difficulties in the traditional treatment of reasoning stem from its separation from the “world”. The effect of this is two-fold:
(1) **Syntactic:** Since the reasoning task is a “stand alone” process, one has to define a rigid representation language with which reasoning problems are presented to the reasoner (Brooks, 1991).

(2) **Performance:** The performance criterion for the behavior of an intelligent system is irrespective of the world it functions in.

In this chapter we embark on developing a new framework for the study of Reasoning. Our approach differs from other approaches to reasoning in that it views learning as an integral part of the process. The *Learning to Reason* theory developed here is concerned with studying the entire process of learning a knowledge representation and reasoning with it.

Our model directly deals with the two problems mentioned above. The first is handled by bypassing the intermediate knowledge representation stage - we do not enforce the agent to keep her knowledge in a specific representation language. The second problem is dealt with by making explicit the dependence of the reasoning performance on the input from the world. This is possible only because the agent interacts with the world when constructing the knowledge representation. Moreover, we take the view that a reasoner need not answer efficiently *all* possible queries, but only those that are “relevant”, or “common”, in a well defined sense. In particular, in this framework the intelligent agent is given access to her favorite learning interface, and is also given a grace period in which she can interact with this interface and construct her representation $KB$ of the world $W$. Her reasoning performance is measured only after this period, when she is presented with queries $\alpha$ from some query language, relevant to the world, and has to answer whether $W$ implies $\alpha$.

In a later chapter we prove the usefulness of the Learning to Reason approach by showing that through interaction with the world, the agent truly gains additional reasoning power. We exhibit several results that do not hold in the traditional setting. First, we develop Learning to Reason algorithms for classes of propositional languages for which no efficient reasoning algorithm exists, when represented as a traditional (formula-based) knowledge base. Second, we exhibit Learning to Reason algorithms for a class of propositional languages that is not known to be learnable in the traditional sense.

Furthermore, an inherent feature of the learning to reason approach is a non-monotonic reasoning behavior it exhibits as a side effect, by using reasoning mistakes to improve the knowledge representation. This desirable phenomenon is hard to formalize when dealing with reasoning systems defined independent of learning. This is discussed in greater detail in Chapter 6.

---

1. We stress that the assumption that an intelligent agent has to keep her knowledge in some representation and use it when she reasons is basic to this framework. We allow the reasoner, however, to choose her own representation and even to use different representations for different tasks.
The Learning to Reason approach should be contrasted with various knowledge compilation studies (Selman and Kautz, 1991; Moses and Tennenholtz, 1993). There, a theory ($KB$) is given to the system designer who is trying to compile it, off line, into a more tractable knowledge representation, to facilitate the answering of future queries. In our approach, a world representation is not known to the agent who is trying to access it via some reasonable interface, acquiring information that allows her to answer queries correctly and efficiently.

This work is similar in spirit to the Neuroidal model developed by Valiant (Valiant, 1994a). The model developed there provides a more comprehensive approach to cognition and, akin to our approach, it views learning as an integral and crucial part of the process. There, the reasoner reasons from a learned knowledge base, a complex circuit, and this can be modeled by our framework. Indeed reasoning in the Neuroidal model shares many properties with the Learning to Reason framework. Our approach in this chapter is different, however, in that in an effort to give a more formal treatment of a reasoner that has learned her knowledge base, we currently restrict our discussion to consistent worlds. (But, see the discussion of this issue in Chapter 6.)

In Section 3.1 we give the necessary background from reasoning and learning theory. We formally define the reasoning task and the type of interface we assume the agent has with the world. In Section 3.2 we present a sampling approach to reasoning and use it to motivate the framework, and in Section 3.3 we define the new Learning to Reason framework. In Section 3.4 we study the relation of the Learning to Reason (L2R) framework to the two existing ones, the traditional reasoning and the traditional learning (learning to classify (L2C)). In Section 3.5 we discuss how existing results from learning and reasoning can be used in the new framework. In particular, we exhibit limitations on combining learning and reasoning algorithms in order to yield learning to reason algorithms. Finally, in Section 3.6 we conclude by re-stating the major ingredients of this framework and describe how we proceed to develop it in the next chapters.

3.1 Preliminaries: Learning and Reasoning

The discussion in this chapter is restricted to propositional languages (or: Boolean functions) and we use the terminology and notations introduced in Section 1.1. Let $\mathcal{F}, \mathcal{Q}$ be two arbitrary classes of representations for Boolean functions.

3.1.1 Reasoning

Within the knowledge-based system approach, knowledge in some representation language is stored in a Knowledge Base ($KB$) that is combined with a reasoning mechanism. As we have seen, various reasoning tasks can be defined within this
framework, but we restrict ourselves here to deduction, and unless we say so explicitly, “reasoning” refers to the deduction task. By deduction, we mean the task of determining whether a query $\alpha$, assumed to capture the situation at hand, is implied from $KB$ (denoted $KB \models \alpha$).

**Definition 3.1.1** An algorithm $A$ is an exact reasoning algorithm for the reasoning problem $(\mathcal{F}, Q)$, if for all $f \in \mathcal{F}$ and for all $\alpha \in Q$, when $A$ is presented with input $(f, \alpha)$, $A$ runs in time polynomial in $n$ and the size of $f$ and $\alpha$, and answers “yes” if and only if $f \models \alpha$.

In the definition above the complexity of the algorithm depends on the size of the input, the formulas $f$ and $\alpha$. For this reason we usually restrict the discussion to a polynomial size representation of the functions considered. In particular, the class $\mathcal{F} = \text{CNF}$ denotes those Boolean functions with a polynomial size CNF, and $\mathcal{F} = \text{CNF} \cap \text{DNF}$ denotes those Boolean functions with a polynomial size CNF, and a polynomial size DNF.

Answering the question $KB \models \alpha$ is equivalent to solving satisfiability for the formula $KB \land \overline{\alpha}$. This implies that if $KB$ is given as a CNF or $\alpha$ is given as a DNF the problem is $NP$-Hard. It is also known that if $KB$ is given as a DNF and $\alpha$ is given as a CNF the problem can be solved in polynomial time, but this representation was less favored than the previous ones possibly because of the belief that the $KB$ should represent a set of rules (that easily translate to a CNF representation but not to a DNF), or since it is more difficult to update the representation in this form. Thus, when $KB$ is given as a CNF, exact reasoning can be done efficiently only when satisfiability can be solved efficiently (e.g., Horn theories, CNF with clauses of length two).

### 3.1.2 Learning to Classify

The formal study of learning, (studied in computational learning theory (Valiant, 1984; Haussler, 1987; Angluin, 1992)), abstracts the problem of inductively learning a concept as the problem of learning a Boolean function, given some access to an oracle that is familiar to some degree with the function. The interpretation is that the function’s value is 1 when the input belongs to the target concept and 0 otherwise. The oracles are used to model the type of interface the learner may have to the world and they vary between learning models according to the amount of information we assume the learner receives about the concept. The learner is given some grace period of interaction with the world. Her performance is measured only after this grace period, when she is presented instances of the target concept and is supposed to classify them correctly (always or “most of the time”, according to the model). Next we describe some oracles that are standard in learning theory, introduce others that are especially suited for Reasoning and define the learning problem.

**Definition 3.1.2** A Membership Query Oracle for a function $f$, denoted $MQ(f)$, is an oracle that when given an input $x \in \{0,1\}^n$ returns $f(x)$. 

48
Definition 3.1.3 An Equivalence Query Oracle for a function $f$, denoted $EQ(f)$, is an oracle that when given as input a function $g$, answer “yes” if and only if $f \equiv g$. If it answers “no” it supplies a counterexample, namely, an $x \in \{0, 1\}^n$ such that $f(x) \neq g(x)$. A counterexample $x$ satisfying $f(x) = 1$ ($f(x) = 0$) is called a positive (negative) counterexample.

Definition 3.1.4 An Example Oracle for a function $f$, with respect to the probability distribution $D$, denoted $EX_D(f)$, is an oracle that when accessed, returns $(x, f(x))$, where $x$ is drawn at random according to $D$.

Next we define entailment oracles, originally defined by Frazier and Pitt (1993). In this case, rather than seeing positive and negative instances of the world, as in $EX_D(f)$ and $MQ(f)$, the agent “learns” the world from examples that are statements that the world entails and does not entail.

Definition 3.1.5 An Entailment Membership Query Oracle for a function $f$, denoted $EnMQ(f)$, is an oracle that when given as input a function $g$ answers “yes” if $f \models g$ and “no” otherwise.

Definition 3.1.6 An Entailment Equivalence Query Oracle for a function $f$, denoted $EnEQ(f)$, is an oracle that when given as input a function $g$, answers “yes” if and only if $f \equiv g$. If it answers “no” it supplies either a negative counterexample, namely, a function $h$ such that $f \nvdash h$ but $g \models h$, or a positive counterexample $h$ such that $f \models h$ but $g \nvdash h$. The function $h$ is restricted to be a Horn disjunction.

We note that these are not all the oracles a learner can use when learning a function. For example, the oracles “incomplete membership queries” (Angluin and Slonim, 1991) “faulty example oracles” (Angluin and Laird, 1988) “malicious example oracles” (Valiant, 1985; Kearns and Li, 1988), “statistical queries oracle” (Kearns, 1993) are also studied in the literature and can be used here as well.

The Reasoning oracles we introduce next model the case where an agent learns from mistakes it makes while reasoning.

Definition 3.1.7 A Reasoning Query Oracle for a function $f$ and a query language $Q$, denoted $RQ(f, Q)$, is an oracle that when accessed performs the following protocol with a learning agent $A$. (1) The oracle picks an arbitrary query $\alpha \in Q$ and returns it to $A$. (2) The agent $A$ answers “yes” or “no” according to her belief with regard to the truth of the statement $f \models \alpha$. (3) If $A$’s answer is correct then the oracle says “correct”. If the answer is wrong the oracle answers “wrong” and in case $f \nvdash \alpha$ it also supplies a counterexample (i.e., $x \in f \setminus \alpha$).

Definition 3.1.8 A Strong Reasoning Query Oracle for a function $f$ and a query language $Q$, denoted $SRQ(f, Q)$, is an oracle that when accessed performs the following protocol with a learning agent $A$. (1) The oracle picks an arbitrary query $\alpha \in Q$ and returns it to $A$. (2) The agent $A$ answers “yes” or “no” according to
her belief with regard to the truth of the statement \( f \models \alpha \). When it answers “no”, \( A \) also returns an assignment \( y \) for which she claims that \( y \in f \setminus \alpha \). (3) If \( A \)’s answer is correct then the oracle says “correct”. If the answer is wrong the oracle answers “wrong” and if \( f \not\models \alpha \) it also supplies a counterexample (i.e. \( x \in f \setminus \alpha \)). In the case that \( f \models \alpha \) the assignment \( y \) serves as a counterexample for the belief of \( A \).

We denote by \( I(f) \) the interface available to the learner when learning \( f \). This can be any subset of the oracles defined above, and might depend on some fixed but arbitrary and unknown distribution \( D \) over the instance space \( \{0,1\}^n \).

We now define various performance criteria for learning algorithms. We distinguish between a “batch” type learning scenario, and an “on-line” scenario. In the “batch” scenario the learning algorithm interfaces with the environment, via \( I(f) \), in order to acquire the skill of labeling future instances. The performance of the algorithm is measured only after some “grace period”, that must be of length polynomial in the size of the concept learned (and also in the quality of its performance), when it is required to predict the value of \( f \) on some other example drawn according to \( D \). The dependency of the algorithm on the size of the concept is not made explicit in the following definitions, assuming that it has some polynomial (in \( n \)) representation. We denote by \( h(x) \) the prediction\(^2\) of the algorithm on the example \( x \in \{0,1\}^n \).

**Definition 3.1.9** An algorithm \( A \) is an Exact Learn to Classify (E-L2C) algorithm for a class of functions \( \mathcal{F} \), if there exists a polynomial \( p() \) such that for all \( f \in \mathcal{F} \), when given access to \( I(f) \), \( A \) runs in time \( p(n) \) and then, given any \( x \in \{0,1\}^n \), takes time \( p(n) \) to predict \( \sigma \) such that \( \sigma = f(x) \).

**Definition 3.1.10** An algorithm \( A \) is a Probably Approximately Correct Learn to Classify (PAC-L2C) algorithm for a class of functions \( \mathcal{F} \), if there exists a polynomial \( p((),) \) such that for all \( f \in \mathcal{F} \), on input \( \epsilon, \delta \) and given access to \( I(f) \), \( A \) runs in time \( p(n,1/\epsilon,1/\delta) \) and then given any \( x \in \{0,1\}^n \), predicts \( h(x) \) in time \( p(n,1/\epsilon,1/\delta) \). \( A \)’s predictions have the property that with probability at least \( 1-\delta \), \( \text{Prob}_{x \in D}[f(x) \neq h(x)] < \epsilon \). The parameter \( \delta \) is called the confidence of the algorithm \( A \) and \( \epsilon \) its accuracy.

In the on-line (or, mistake-bound) scenario, algorithm \( A \) is presented with a sequence of examples in \( \{0,1\}^n \). At each stage, the algorithm is asked to predict \( f(x) \) and is then told whether its prediction was correct. Each time the learning algorithm makes an incorrect prediction, we charge it one mistake.

**Definition 3.1.11** An algorithm \( A \) is a Mistake Bound Learn to Classify (MB-L2C) algorithm for a class of functions \( \mathcal{F} \), if there exists a polynomial \( p() \) such that for all \( f \in \mathcal{F} \), for every (arbitrary infinite) sequence of instances, \( A \) runs in time \( p(n) \) (on each example) and makes no more than \( p(n) \) mistakes.

\(^2\)Sometimes an equivalent definition is used, in which the learning algorithm outputs an hypothesis \( h \) and its performance is measured with respect to it. In order not to make any assumptions on how this hypothesis is represented we assume here that it is internal to the algorithm.
It is known (Angluin, 1988; Littlestone, 1989) that a mistake bound learning algorithm can be transformed into a pac learning algorithm, and that an E-L2C (or PAC-L2C) algorithm that uses the $EQ(f)$ oracle can be transformed into an algorithm that achieves PAC-L2C using an Example Oracle and without using $EQ(f)$.

### 3.2 Motivation: A Sampling Approach

In this section we formally present the example discussed earlier in this chapter and use it to motivate the Learning to Reason model. Rather than presenting it as a question of computing degree of belief we make use of the notion of $\epsilon$-fair queries, and present it as a question of exact deduction, the reasoning task we discuss later. It will be clear that the same argument can be used in both cases.

Let $W \in \mathcal{F}$ be a Boolean function that describes the world exactly. Let $\alpha$ be some Boolean function (a query) and let $D$ be some fixed but arbitrary and unknown probability distribution over the instance space $\{0, 1\}^n$. As in the learning framework, we assume that $D$ governs the occurrences of instances in the world.

**Definition 3.2.1** The query $\alpha$ is called legal if $\alpha \in Q$.

**Definition 3.2.2** The query $\alpha$ is called $(W, \epsilon)$-fair if either $W \subseteq \alpha$ or $\text{Prob}_D[W \setminus \alpha] > \epsilon$.

The intuition behind this definition is that the algorithm is allowed to err in case $W \not\subseteq \alpha$, but the weight of $W$ outside $\alpha$ is very small. Along with $\epsilon$, the accuracy parameter, we use a confidence parameter, $\delta$, and sometimes might allow the reasoning algorithm to err, with small probability, less than $\delta$.

Consider the following simple approach to reasoning: whenever presented with a query $\alpha$, first use the Example Oracle $EX_D(W)$ and take a sample of size $m = (1/\epsilon) \ln(1/\delta)$, where $\delta$ and $\epsilon$ are the required confidence and accuracy parameters. Then, perform the following model-based test: for all the samples $(x, 1)$ sampled from $EX_D(W)$ (note that we ignore the samples labeled 0), check whether $\alpha(x) = 1$. If for some $x$, $\alpha(x) = 0$ say $W \not\models \alpha$; otherwise say $W \models \alpha$.

The following analysis shows that if $\alpha$ is $(W, \epsilon)$-fair then with probability at least $1 - \delta$ the algorithm is correct.

Clearly, if $W \models \alpha$ the algorithm never errs. The algorithm makes a mistake only if $W \not\models \alpha$ but an instance $x$ in $W \cap \overline{\alpha}$ is never sampled. An instance $x$, picked randomly according by $EX_D(W)$ is in $W \cap \overline{\alpha}$ with probability greater than $\epsilon$. Therefore, the probability that an instance $x \in W \cap \overline{\alpha}$ is missed in $m = \frac{1}{\epsilon} \ln(1/\delta)$ trials is less than

$$ (1 - \epsilon)^m = (1 - \epsilon)^{\frac{1}{\epsilon} \ln(1/\delta)} < \delta. $$

Therefore, the algorithm errs on a $(W, \epsilon)$-fair query with probability less than $\delta$.

This analysis depends on the fact that the samples are independent of the query $\alpha$, and therefore a different sample has to be taken for every query $\alpha$. We call
this a repeated sampling approach. However, repeated sampling is not a plausible approach to reasoning in intelligent systems. When presented with a query, an agent cannot allow herself further interactions with the world before answering the query. Especially if the query is “A lion is approaching ⇒ I have to run away”.

For a more plausible approach, we now modify the above procedure and analysis and show that a one time sampling approach can also guarantee reasoning with respect to \((W, \epsilon)\)-fair queries, with confidence \(1 - \delta\). The procedure we use is the following:

Algorithm S-Reason:

Test Set: A set \(S\) of \(m\) samples collected using \(EX_D(W)\).

Test: Given a query \(\alpha\), test if there is an element \(x \in S\) which satisfies \(W\) (that is, a positive sample in \(S\)), but does not satisfy \(\alpha\). In this case, deduce that \(W \not\models \alpha\); Otherwise, \(W \models \alpha\).

Figure 3.1: S-Reason: A Sampling approach to Reasoning

From Eq.3.1 we have that the probability that any \((W, \epsilon)\)-fair query \(\alpha \in Q\), that is not implied by \(W\) is answered by “yes” after \(m\) samples is less than

\[
|Q|(1 - \epsilon)^m, \tag{3.2}
\]

where \(|Q|\) is the size of the class \(Q\) of queries. Therefore, if we take more than

\[
m = \frac{1}{\epsilon}(\ln |Q| + \ln \frac{1}{\delta}) \tag{3.3}
\]

samples from \(EX_D(f)\), by substituting \(m\) from Eq. 3.3 in Eq. 3.2 we have that the model-based test described above errs on every \((W, \epsilon)\)-fair query with probability less than \(\delta\). Since all the queries in \(Q\) are propositional formulas of polynomial size, the number \(m\) of samples required to guarantee this performance is polynomial. This approach is therefore feasible. To summarize, we have:

**Theorem 3.2.1** Let \(0 < \epsilon, \delta < 1\) be given and let \(m = \frac{1}{\epsilon}(\ln |Q| + \ln \frac{1}{\delta})\) be the number of samples taken. Then, for every world \(W\) and every \((W, \epsilon)\)-fair query \(\alpha\), with probability \(> 1 - \delta\), the procedure S-Reason answers \(W \models \alpha\) correctly. In particular, since the number of samples required is polynomial, the procedure is polynomial.

It is important not to confuse this approach of sampling from \(D\), the distribution that governs the occurrences of instances in the world, with sampling approaches

---

3A similar, more sophisticated approach was developed in (Kearns, 1992) for the case in which both the knowledge base and the queries are learned concepts in the pac sense. It is implicit there that for each possible query one needs a new sample.
that aim at performing approximate reasoning from a fixed knowledge base (e.g., a formula-based knowledge base \( W \)). There, no access to the “world” \( EXD(W) \) is available, and the sampling is done, in general, from the uniform distribution, in order to find satisfying assignments of \( W \). This approach, that is typical in many approximate reasoning techniques, was discussed in the previous chapter and was shown to be intractable in a very strong sense.

The new approach exemplifies the power gained by giving the reasoner access to the “world” she is supposed to reason in later. Learning to reason is possible for arbitrary world and query languages given that the queries are fair. We note, that the above algorithm highlights another difference between learning to reason and learning to classify. In the learning to classify approach the above argument yields the result on “Occam” algorithms (Blumer et al., 1987). Namely, if you take a big enough sample and find a small enough consistent hypothesis then you have learned to classify. Unfortunately, in many cases, the problem of finding a consistent hypothesis is hard. For the learning to reason approach, the above argument shows that this hard task is not necessary.

However, the one-time sampling approach is not the ultimate solution to reasoning and there are many reasons not to be satisfied with this approach as the sole solution for the reasoning problem. The most significant caveats in this approach are the assumption it makes, that the samples taken are independent, an assumption that we do not believe to hold too often in practice, and the technical restriction of \( e \)-fair queries, which is not well understood. We briefly discuss some more reasons as a motivation for the approach taken in the rest of the thesis.

**Exact reasoning:** In many cases it is natural to require the reasoning to exactly reflect the agent’s current knowledge. Later in the thesis we show that indeed, unlike the sampling approach, other representations can support it.

**Active learning:** Certain events we want to reason about might be too rare to have a significant weight under \( D \), the distribution that governs the occurrences of instances in the world (i.e., they might not be “fair”). To facilitate this, we might want to supply the reasoner with a richer class of interfaces with the world. Oracles that seem to be useful for that are example oracles that supply examples restricted to satisfy a specific context (e.g., “I am in Boston”), partial assignments oracles and more selective oracles (Amsterdam, 1988). We show later, in Chapter 4, how our study of exact reasoning applies also to reasoning within varying context, and in Chapter 6 we extend it to discuss partial assignment oracles.

**Incremental/Non-monotonic nature of Reasoning:** We present later in this thesis algorithms for reasoning from an inductively learned knowledge base which, unlike reasoning from a large set of randomly selected examples, converge, while exhibiting desirable a non-monotonic behavior, to yield an exact reasoning behavior. We will see that “on-line” algorithms are better suited for
our purpose, and will show that the on-line algorithms and the representations we develop to study exact reasoning support incremental reasoning.

### 3.3 Learning to Reason: Definitions

We now provide the basic definitions that allow us to initiate the formal study into the questions discussed above. As pointed out above, our definitions do not allow a repeated sampling approach, but do allow one-time sampling.

As in the case of Learning to Classify we distinguish between Learning to Reason in a “batch” type scenario, and an “on-line” Learning to Reason.

In the batch scenario, the algorithm interfaces with the environment, via \(I(f)\), in order to acquire the reasoning skill for answering future queries. The performance of the algorithm is measured only after some “grace period”, that must be of length polynomial in the size of the world description (and in the quality of its performance), when it is required to decide, without interfacing the world again, if some query is entailed by \(f\). As before we define an exact version and a non-exact version.

**Definition 3.3.1** An algorithm \(A\) is an Exact Learn to Reason (E-L2R) algorithm for the reasoning problem \((\mathcal{F}, \mathcal{Q})\), if there exists a polynomial \(p()\) such that for all \(f \in \mathcal{F}\), given access to \(I(f)\), \(A\) runs in time \(p(n)\) and then, when presented with any query \(\alpha \in \mathcal{Q}\), \(A\) runs in time \(p(n)\), does not access \(I(f)\), and answers “yes” if and only if \(f \models \alpha\).

**Definition 3.3.2** An algorithm \(A\) is a Probably Approximately Correct Learn to Reason (PAC-L2R) algorithm for the reasoning problem \((\mathcal{F}, \mathcal{Q})\), if there exists a polynomial \(p(), ()\) such that for all \(f \in \mathcal{F}\), on input \(\epsilon, \delta\), given access to \(I(f)\), \(A\) runs in time \(p(n, 1/\epsilon, 1/\delta)\) and then with probability at least \(1 - \delta\), when presented with any \((f, \epsilon, \delta)\)-fair query \(\alpha \in \mathcal{Q}\), \(A\) runs in time \(p(n, 1/\epsilon, 1/\delta)\), does not access \(I(f)\), and answers “yes” if and only if \(f \models \alpha\).

In the batch scenario above we did not allow access to \(I(f)\) while in the query answering phase. In the on-line version however, we consider a query \(\alpha\) given to the algorithm as if given by the reasoning oracle \(RQ(f, \mathcal{Q})\) defined above. Thus, a reasoning error may supply the algorithm a counterexample which in turn can be used to improve its future reasoning behavior. We allow the L2R algorithm to access \(I(f)\) during this update, but not while answering a query. In this on-line (or, mistake-bound) scenario, the L2R algorithm is charged one mistake each time the reasoning query is answered incorrectly.

**Definition 3.3.3** An algorithm \(A\) is a Mistake Bound Learn to Reason (MB-L2R) algorithm for the reasoning problem \((\mathcal{F}, \mathcal{Q})\), if \(A\) interacts with the reasoning oracle \(RQ(f, \mathcal{Q})\), and there exists a polynomial \(p()\) such that for all \(f \in \mathcal{F}\), \(1)\) \(A\) runs in time \(p(n)\) (on each query) and answers “yes” or “no” according to its belief with
regard to the truth of the statement \( f \models \alpha \), without accessing \( I(f) \), (2) then runs in time \( p(n) \) before it is ready for the next query (possibly, with accessing \( I(f) \)), and (3) for every (arbitrary infinite) sequence of queries, \( A \) makes no more than \( p(n) \) mistakes.

### 3.4 The relations between L2R and L2C

In this section we investigate the relation between Learning to Reason and Learning to Classify. Clearly, given an Exact-L2C algorithm \( A \), the ability to reason (in the traditional sense) efficiently with the hypothesis \( A \) keeps is sufficient to yield an efficient Exact L2R algorithm. In this section we consider the other direction of this relation. That is, given an algorithm that can Learn to Reason, is it necessarily the case that there is an algorithm that can Learn to Classify?

Intuitively, the classification task seems to be easier than the reasoning task. In the former, we need to evaluate correctly a function on a single point, while in the latter we need to know if all the models of the function are also models of the query. It is not surprising therefore, that if any subset of \( \{0, 1\}^n \) is a legal query, the ability to L2R implies the ability to L2C. We formalize this in the following theorem.

Let \( \text{DISJ} \) be the class of all disjunctions over \( n \) variables.

**Theorem 3.4.1** If there is an Exact-L2R algorithm for the reasoning problem \( (\mathcal{F}, \text{DISJ}) \) then there is an Exact-L2C algorithm for the class \( \mathcal{F} \).

**Proof:** Observe that for a Boolean function \( f \) and an assignment \( z \in \{0, 1\}^n \),

\[
    f(z) = 0 \iff f \models \{0, 1\}^n \setminus \{z\}. \quad (3.4)
\]

Denote by \( d_z \) the disjunction defined by \( d_z = \lor x_i^z \) (where \( x_i^0 = x_i, x_i^1 = \overline{x_i} \)). For example, if \( z = (1, 0, 0) \) then \( d_z = \overline{x_1} \lor x_2 \lor x_3 \). Clearly \( d_z \) satisfies \( d_z(y) = 0 \) if and only if \( y = z \), and Eq. 3.4 can be written therefore as

\[
    f(z) = 0 \iff f \models d_z. \quad (3.5)
\]

Now, given an Exact-L2R algorithm \( A \) for the reasoning problem \( (\mathcal{F}, \text{DISJ}) \) we use it to construct an Exact-L2C algorithm \( B \) for \( \mathcal{F} \) as follows: \( B \) first runs \( A \) (enough time to guarantee \( A \)'s performance), and when given an assignment \( z \in \{0, 1\}^n \) for classification by \( f \), it first computes the disjunction \( d_z \) and gives it to \( A \). \( B \) returns “1” if and only if \( A \) returns “no” on \( d_z \). The correctness of the algorithm is clear from Eq. 3.5.

We note that reasoning with DISJ is as hard as reasoning with general CNF queries, since in general \( f \models (\alpha \land \beta) \) if and only if \( f \models \alpha \) and \( f \models \beta \).

The proof of the above theorem does not go through if the class of queries \( Q \) does not include all of \( \text{DISJ} \), and does not hold also in the non-exact L2R formulation. Strictly speaking, we leave open the question of whether there is a case, in which
there exists a Learning to Reason algorithm for the problem \((\mathcal{F}, \mathcal{Q})\) but there is no Learning to Classify algorithm for \(\mathcal{F}\). The reason is that in general, it is hard to prove that there does not exist a \(\text{L}2\text{C}\) algorithm. We can show however, that there is a class of Boolean functions \(\mathcal{F}\), and a class of queries \(\mathcal{Q}\), such that there exists a Learning to Reason algorithm for the problem \((\mathcal{F}, \mathcal{Q})\) but it is not known how to Learn to Classify \(\mathcal{F}\). We defer the presentation of this result to a later chapter (Section 5.2).

### 3.5 L2R by Combining Learning and Reasoning

In this section we identify techniques for Learning to Reason. We show how the interaction with the world (i.e., learning) can be used to collect some approximate knowledge, and then how to use this knowledge to reason successfully. The results in this section combine results from computational learning theory with results from the theory of reasoning, and thus can be considered as a positive outcome of studying these problems separately. However, as we show below, the significance of the results presented in this section is that they exhibit the limitations of L2R by combining separate reasoning and learning algorithms.

The case of exact learning algorithms that, in addition, produce as output a representation that allows for efficient reasoning has already been discussed. Here we consider the relaxation of these requirements.

#### 3.5.1 Learning to Reason via PAC Learning

Assume that the world description \(W\) is in \(\mathcal{F}\) and there is a PAC-L2C algorithm \(A\) for \(\mathcal{F}\). We first show how a pac learning algorithm, if it has an additional property, can be combined with a reasoning algorithm to yield a PAC-Learn to Reason algorithm.

**Definition 3.5.1** An algorithm that pac learns to classify \(\mathcal{F}\) is said to learn \(f \in \mathcal{F}\) from below if, when learning \(f\), the algorithm never makes mistakes on instances outside of \(f\). (I.e., if \(h\) is the hypothesis the algorithm keeps then it satisfies \(h \subseteq f\).)

**Theorem 3.5.1** Let \(A\) be a PAC-Learn to Classify algorithm for the function class \(\mathcal{F}\) and assume that \(A\) uses the class of representations \(\mathcal{H}\) as its hypotheses. Then, if \(A\) learns \(\mathcal{F}\) from below, and there is an exact reasoning algorithm \(B\) for the reasoning problem \((\mathcal{H}, \mathcal{Q})\), then there is a PAC-Learn to Reason algorithm \(C\) for the reasoning problem \((\mathcal{F}, \mathcal{Q})\).

**Proof:** The learning algorithm \(C\) simply runs \(A\) and then uses \(B\) in order to reason with the hypothesis \(h \in \mathcal{H}\) of \(A\). Let \(\alpha\) be a \((W, \epsilon)\)-fair legal query. Assume first that \(W \models \alpha\). Then, since the algorithm \(A\) learns \(W\) from below its hypothesis \(h\) satisfies \(h \subseteq W \subseteq \alpha\) and therefore \(h \models \alpha\) and \(C\) answers correctly. When \(W \not\models \alpha\), since \(\alpha\) is \((W, \epsilon)\)-fair, we know that \(\text{Prob}_D[W \setminus \alpha] > \epsilon\). Together with the fact that
\[ \Pr_{\mathcal{D}}[W \setminus h] < \epsilon, \] we have \( h \setminus \alpha \neq \emptyset \) and therefore \( h \not\models \alpha \), so again, \( C \) answers correctly.

The performance guaranteed by the above theorem is not better than the one-time sampling approach, while using a possibly more complicated algorithm. The result is significant, however, for the following reasons: (1) It shows the limitations of L2R by combining reasoning and learning algorithms: The theorem cannot be extended to the case where the the learning algorithm has also error from above (i.e., where it makes prediction mistakes also on instances of \( W \)). The reason is that if \( h \setminus W \neq \emptyset \) then it might be the case that \( W \models \alpha \) but \( h \not\models \alpha \) exactly because of those additional assignments in \( h \). (2) The theorem allows for the pac learning algorithms to use any set of oracles (and in particular, other, more plausible interfaces to specific applications) while the sampling result holds only for the example oracle. (3) The theorem allows for a system that performs classification as well as reasoning from the same knowledge base, and finally, (4) The theorem explains the behavior of mistake bound algorithms discussed in the next section.

For a similar result to hold for an algorithm with two sided error one has to use a somewhat strong version of an “approximate deduction”.

**Definition 3.5.2** An algorithm \( A \) is an approximate deduction algorithm for the reasoning problem \((\mathcal{F},\mathcal{Q})\), if for every \( \epsilon \), and for all \( f \in \mathcal{F} \) and \( \alpha \in \mathcal{Q} \) when presented with input \((f,\alpha)\), \( A \) runs in time polynomial in \( n \) and \( 1/\epsilon \) and answers “yes” if and only if \( \Pr_{\mathcal{D}}[f \setminus \alpha] < \epsilon \).

Given this notion of deduction, we can now show:

**Theorem 3.5.2** Let \( A \) be a PAC-Learn to Classify algorithm for the function class \( \mathcal{F} \) and assume that \( A \) uses the class of representations \( \mathcal{H} \) as its hypotheses. If there is an approximate deduction algorithm \( B \) for the reasoning problem \((\mathcal{H},\mathcal{Q})\), then there is a PAC-Learn to Reason algorithm \( C \) for the reasoning problem \((\mathcal{F},\mathcal{Q})\).

**Proof:** The algorithm \( C \) runs \( A \), enough time to guarantee an \( \epsilon \)-accurate hypothesis \( h \). Then, given a query \( \alpha \), it uses the \( \epsilon \)-approximate deduction algorithm \( B \). We show that \( C \) succeeds on \((W,2\epsilon)\)-fair queries.

Suppose \( W \models \alpha \). Then \( \Pr_{\mathcal{D}}[W \setminus \alpha] = 0 \), which implies \( \Pr_{\mathcal{D}}[h \setminus \alpha] < \epsilon \) and the algorithm answers correctly, “yes”.

In the other case, when \( W \not\models \alpha \), since \( \alpha \) is \((W,2\epsilon)\)-fair, we know that \( \Pr_{\mathcal{D}}[W \setminus \alpha] > 2\epsilon \). Therefore \( \Pr_{\mathcal{D}}[h \setminus \alpha] > \epsilon \) and the algorithm answers correctly, “no”. ■

### 3.5.2 Learning to Reason via Mistake Bound Learning

Consider a Mistake Bound algorithm that keeps a hypothesis that allows for efficient reasoning. In this section we observe that in this case, the algorithm can be used to construct a Learn to Reason algorithm.

Let \( A \) be a mistake bound algorithm and assume it has been used long enough to guarantee pac performance (Littlestone, 1989). In the case it has used up all of its
mistakes on negative examples (i.e., on assignments outside of $W$), the hypothesis it uses is a “learn from below” hypothesis, and we can reason with it and succeed on all $(W, c)$-fair queries.

Unfortunately, we cannot force the algorithm (or rather the interface) to make all these mistakes within the grace period. If we use an initial grace period to ensure its pac properties then after the algorithm is ready to answer queries it may still make (a limited number of) mistakes. If the reasoning mistakes can be used as a source for negative counterexamples (i.e., assignments in $h \setminus W$) then after making a polynomial number of reasoning mistakes the algorithm would hold a hypothesis that approximates $W$ from below, and thus supports exact reasoning.

This is the behavior we model with the reasoning oracle, $RQ(f, Q)$. Notice though, that while this is enough to guarantee exact reasoning, it is not enough to learn even a pac approximation of $W$, since the algorithm receives only positive counterexamples. Using the strong oracle $SRQ(f, Q)$ instead, the algorithm is allowed to use also negative counterexamples and in this case the hypothesis will converge to an exact description of $W$.

It is interesting to note that reasoning with this type of an algorithm yields a non monotonic reasoning behavior. Every time the algorithm makes a reasoning mistake, it changes its mind, learns something about the world, and would not make the same mistake again. This is an inherent feature of the learning to reason approach, and it captures a phenomenon that is hard to formalize, when dealing with reasoning systems defined independent of learning.

### 3.6 Concluding Remarks

We have motivated and defined the new framework for the study of reasoning in intelligent systems. The Learning to Reason approach was defined in a way that is aimed at overcoming the main computational difficulties in the traditional treatment of reasoning.

This framework differs from existing ones in that it views learning as an integral part of the process and in particular, it allows the agent to interact with world, in the spirit of the known learning models. By allowing the reasoning task to interact with the world we avoid the rigid syntactic restrictions imposed in the knowledge-based systems framework on the intermediate knowledge representation. At the same time, the framework makes explicit the dependence of the reasoning performance on the input from the environment.

We have discussed the relation of the new framework to the known computational approaches to learning and to reasoning, and have shown how previous results in learning and reasoning can be used in the new framework. While reasoning from an inductively learned hypothesis is a desirable goal, there are two subtle issues that prevent a direct integration of results from Learning theory and Reasoning. First, there is the obvious problem that the traditional approach to reasoning struggles with: it is important that the output of the learning phase is presented in a form that
is amenable to efficient solutions of the prescribed task. The second issue concerns using learning procedures that output only “approximate” representations. For example, pac learning has been accepted as a good measure of learning even when learning for the purpose of performing reasoning (e.g. when learning logic programs (Cohen, 1994)). As we have observed in this chapter (and see also (Kearns, 1992)) learning algorithms with guaranteed pac performance may yield erroneous reasoning behavior unless they have an additional property: the hypothesis $K_B$ must be a subset of the function $W$ (or at least a subset of its least upper bound). Therefore, when using pac learning algorithms for the purpose of performing deductive reasoning tasks, this additional property must be imposed.

So far, due to the limitations we have observed on the possibility of directly integrating learning and reasoning algorithms, we have gained only few positive results from the new framework. The severe restrictions on efficient reasoning from formula-based knowledge representations imply that in order to make some progress in performing deduction, we will have to consider different knowledge representations. We do that in the next chapter.
Bohr never trusted purely formal mathematical argument. “No, no” he would say, “You are not thinking; you are just being logical.”

O. R. Frish, What Little I Remember, 1979

In previous chapters we have introduced the knowledge-based system approach to reasoning. There, knowledge is stored in some representation language with a well defined meaning, and the emphasis is on the comprehensibility\(^1\) of the stored knowledge (McCarthy and Hayes, 1969). Levesque (1986; 1992) argues that reasoning with a more direct representation is easier and better suits commonsense reasoning. He suggests to represent the knowledge base \(KB\) in a vivid form, one that bears a strong and direct relationship to the real world. This might be just a model of \(KB\) (Etherington et al., 1989; Papadimitriou, 1991) on which one can evaluate the truth value of the query \(\alpha\). It is not clear, however, how one might derive a vivid form of the knowledge base, and as expected, selecting a model that is, informally, the most likely model of the real world based on any reasonable criterion is a computationally hard task (Papadimitriou, 1991; Selman and Kautz, 1990). Most importantly, in order to achieve an efficient solution to the reasoning problem this approach modifies the problem: reasoning with a vivid representation no longer solves the problem \(KB \vdash \alpha\), but rather a different problem, whose exact relation to the original inference problem depends on the method selected to simplify the knowledge base. (See also the discussion in Section 6.3.1 of the problem of selecting a model.)

In this chapter we develop a model-based approach to commonsense reasoning, in which the knowledge base is represented as a set of models (satisfying assignments)

\(^1\)This emphasis is true not only for the logical, but also for probabilistic approaches to reasoning, e.g., (Pearl, 1988), but we restrict ourselves here to the logical approach.
of the world rather than a logical formula describing it. It is not hard to motivate a model-based approach to reasoning from a cognitive point of view and indeed, most of the proponents of this approach to reasoning have been cognitive psychologists (Johnson-Laird, 1983; Johnson-Laird and Byrne, 1991; Kosslyn, 1983), who have alluded to the notion of “reasoning from examples” on a qualitative basis. In the AI community this approach can be seen as an example of Levesque’s notion of “vivid” reasoning mentioned above, is somewhat related to Minsky’s frames-theory (Minsky, 1975) and has already been studied in a narrower context in (Kautz, Kearns, and Selman, 1993).

The problem $KB \models \alpha$ can be approached using the following model-based strategy:

**Algorithm MBR:**

Test Set: A set $S$ of possible assignments.

Test: If there is an element $x \in S$ which satisfies $KB$, but does not satisfy $\alpha$, deduce that $KB \not\models \alpha$; Otherwise, $KB \models \alpha$.

Figure 4.1: MBR: Model-Based Reasoning

Clearly, this approach solves the inference problem if $S$ is the set of all models (satisfying assignments) of $KB$, but this set might be too large. A model-based approach becomes useful if one can show that it is possible to use a fairly small set of models as the Test Set, and still perform reasonably good inference, under some criterion.

We define a set of models, the *characteristic models* \(^2\) of the knowledge base, and show that performing the model-based test with it suffices to deduce that $KB \models \alpha$, for a restricted set of queries. We prove that for a fairly wide class of representations, this set is sufficiently small, and thus the model-based approach is feasible.

In this chapter we formally introduce the notion of restricted queries, which we have motivated in Chapter 2 and is inherent to our approach; since we are interested in formalizing commonsense reasoning, we take the view that a reasoner need not answer efficiently all possible queries, but rather to answer restricted classes of queries with respect to a complicated “world”. Moreover, we prove an interesting and useful result about reasoning with restricted queries: when reasoning with a wide, but restricted, class of queries it is sufficient that the knowledge base holds only an approximate description of the world (in a well defined sense) in order to reason exactly about it. This result is used to enlarge the class of worlds for which

\(^2\)We note that characteristic models were studied independently in the Relational Data Base community (where they are called “generators”) (Beeri et al., 1984; Mannila and Raiha, 1986), for the special case of definite Horn theories. Our results have some immediate implications in this domain which we develop elsewhere (Khardon, Mannila, and Roth, 1994).
model-based reasoning is feasible. This view can be contrasted with an orthogonal line of research, which aims at identifying classes of worlds of limited expressiveness with which one can perform theorem proving efficiently with respect to all queries. We comment that this approach of “restricting the world” has been criticized also on the grounds that existing results do not meet the strong tractability requirements for commonsense reasoning as described, for example, in (Shastri, 1993), even though (as argued, for example, in (Doyle and Patil, 1991)) the inference deals with limited expressiveness and is sometimes restricted in implausible ways.

The main interest in model-based representation arises from its computational efficiency. We show that using the model-based representation developed here, we can solve efficiently many reasoning problems which are known to be intractable using the traditional representation. For example, let \( KB \) be a CNF formula, whose DNF\(^3 \) size is polynomial, and \( \alpha \) a query in \( \log n \)CNF. Then, solving the problem \( KB \models \alpha \) is co-NP-hard, but \( KB \) has a polynomial size model-based representation that can be used to answers all queries \( \alpha \) efficiently. Clearly, our algorithms do not solve NP-complete problems. Most hardness results for reasoning assume that \( KB \) is given as a CNF formula. The fact that we can perform reasoning efficiently relies on that we change the knowledge representation into a more accessible form. We will appreciate the usefulness of this approach even more in the next chapter, when we embed it within the Learning to Reason process and show that the model-based representations studied here are efficiently learnable.

Equally important, we believe, is the view that model-based representations suggest about reasoning, and in particular, logical reasoning. First, we show that reasoning with model-based representations enjoy some desirable robustness properties, which logical representations lack, and that they support incremental reasoning. Moreover, the results illustrate and aid in formalizing the notion of “intelligence is in the eye of the beholder” (Brooks, 1991). Using a model-based representation, our agent behaves logically, even though her knowledge representation consists of a set of models and not a logical formula and she does not use any logic or “theorem proving”.

In Section 4.1 we introduce some notations and present the monotone theory, a characterization of Boolean functions introduced in (Bshouty, 1993a), which we use in many of our technical results.

In the main section of this chapter, Section 4.2, we consider the deduction problem. We introduce the set of characteristic models and analyze the correctness and efficiency of model-based deduction with this set. We use characteristic models to represent approximate theories and characterize the situations in which reasoning

---

\(^3\)The size of the model-based representation of \( KB \) is related to the size of its minimal DNF. Thus, we do not assume that the DNF representation is known but only require that a polynomial size representation exists.
with approximate theories yield exact deduction. We then prove the necessity of a complete set of characteristic models to performing exact deduction.

In Section 4.3 we show that in the special case of Horn theories our theory reduces to the work in (Kautz, Kearns, and Selman, 1993).

The complexity of model-based reasoning is directly related to the number of models in the representation. This issue is investigated in Section 4.4, where we compare this size of the model based representation of a Boolean function with the size of other representations of the same function. In particular, our results characterize the Horn theories for which the approach suggested in (Kautz, Kearns, and Selman, 1993) is useful and explain the phenomena observed there, regarding the relative sizes of the logical formula representation and model-based representation of $KB$.

In Section 4.5 we discuss applications of the our theory to particular propositional languages, other than Horn, and summarize the results on model-based deduction. We also show how the techniques presented here can be used to solve an open question about structure identification (Dechter and Pearl, 1992).

In Section 4.6 we consider the problem of performing abduction using a model-based approach and show that for any propositional knowledge base, using a model-based representation yields an abductive explanation in time that is polynomial in the size of the representation.

In Section 4.7 we consider some important robustness issues of a model-based representation. In particular, we consider the problem of augmenting a model-based representation with a set of rules and the problem of reasoning within varying context.

Finally, in Section 4.8 we conclude by showing how model-based representations support an incremental view of reasoning.

## 4.1 Monotone Theory

In this section we introduce the notations, definitions and results of the Monotone Theory of Boolean functions, introduced by Bshouty (Bshouty, 1993a).

**Definition 4.1.1 (Order)** We denote by $\leq$ the usual partial order on the lattice $\{0,1\}^n$, the one induced by the order $0 < 1$. That is, for $x, y \in \{0,1\}^n$, $x \leq y$ if and only if $\forall i, x_i \leq y_i$. For an assignment $b \in \{0,1\}^n$ we define $x \leq_b y$ if and only if $x \oplus b \leq y \oplus b$ (where $\oplus$ is the bitwise addition modulo 2).

Intuitively, if $b_i = 0$ then the order relation on the $i$th bit is the normal order; if $b_i = 1$, the order relation is reversed and we have that $1 \leq_b 0$.

The monotone extension of $z \in \{0,1\}^n$ with respect to $b$ is:

$$\mathcal{M}_b(z) = \{x \mid x \geq_b z \}. $$
The monotone extension of \( f \) with respect to \( b \) is:

\[
\mathcal{M}_b(f) = \{ x \mid x \geq_b z, \text{ for some } z \in f \}.
\]

The set of minimal assignments of \( f \) with respect to \( b \) is:

\[
\min_b(f) = \{ z \mid z \in f, \text{ such that } \forall y \in f, z \not<_b y \}.
\]

The following claim lists some properties of \( \mathcal{M}_b \), all of which are immediate from the definitions:

**Claim 4.1.1** Let \( f, g : \{0, 1\}^n \rightarrow \{0, 1\} \) be Boolean functions. The operator \( \mathcal{M}_b \) satisfies the following properties:

1. If \( f \leq g \) then \( \mathcal{M}_b(f) \subseteq \mathcal{M}_b(g) \).
2. \( \mathcal{M}_b(f \land g) \subseteq \mathcal{M}_b(f) \land \mathcal{M}_b(g) \).
3. \( \mathcal{M}_b(f \lor g) = \mathcal{M}_b(f) \lor \mathcal{M}_b(g) \).
4. \( f \subseteq \mathcal{M}_b(f) \).

**Claim 4.1.2** Let \( z \in f \). Then, for every \( b \in \{0, 1\}^n \), there exists \( u \in \min_b(f) \) such that \( \mathcal{M}_b(z) \subseteq \mathcal{M}_b(u) \).

**Proof:** If \( z \not\in \min_b(f) \) then \( \exists y \in f \) such that \( y \leq_b z \). Let \( u \) be a minimal element in \( f \) with this property. Then, \( u \in \min_b(f) \) and clearly \( \{ x \mid x \geq_b z \} \subseteq \{ x \mid x \geq_b u \} \), as needed.

Using Claims 4.1.2 and 4.1.1 we get a characterization of the monotone extension of \( f \):

**Claim 4.1.3** The monotone extension of \( f \) with respect to \( b \) is:

\[
\mathcal{M}_b(f) = \bigvee_{z \in f} \mathcal{M}_b(z) = \bigvee_{z \in \min_b(f)} \mathcal{M}_b(z).
\]

Clearly, for every assignment \( b \in \{0, 1\}^n \), \( f \subseteq \mathcal{M}_b(f) \). Moreover, if \( b \not\in f \), then \( b \not\in \mathcal{M}_b(f) \) (since \( b \) is the smallest assignment with respect to the order \( \leq_b \)). Therefore:

\[
f = \bigwedge_{b \in \{0, 1\}^n} \mathcal{M}_b(f) = \bigwedge_{b \not\in f} \mathcal{M}_b(f).
\]

The question is if we can find a small set of negative examples \( b \), and use it to represent \( f \) as above.

**Definition 4.1.2 (Basis)** A set \( B \) is a basis for \( f \) if \( f = \bigwedge_{b \in B} \mathcal{M}_b(f) \). \( B \) is a basis for a class of functions \( F \) if it is a basis for all the functions in \( F \).

Using this definition, the representation

\[
f = \bigwedge_{b \in B} \mathcal{M}_b(f) = \bigwedge_{b \in B} \bigvee_{z \in \min_b(f)} \mathcal{M}_b(z)
\]  

(4.1)
yields the following necessary and sufficient condition describing when \( x \in \{0, 1\}^n \) is positive for \( f \):

**Corollary 4.1.4** Let \( B \) be a basis for \( f \), \( x \in \{0, 1\}^n \). Then, \( x \in f \) (i.e., \( f(x) = 1 \)) if and only if for every basis element \( b \in B \) there exists \( z \in \min_b(f) \) such that \( x \geq_b z \).

The following claim bounds the size of the basis of a function \( f \):

**Claim 4.1.5** Let \( f = C_1 \land C_2 \land \cdots \land C_k \) be a CNF representation for \( f \) and let \( B \) be a set of assignments in \( \{0, 1\}^n \). If every clause \( C_i \) is falsified by some \( b \in B \) then \( B \) is a basis for \( f \). In particular, \( f \) has a basis of size \( \leq k \).

**Proof:** Let \( B = \{b^1, b^2, \ldots, b^k\} \) be a collection of assignments such that \( b^i \) falsifies \( C_i \). We show that \( f \equiv \bigwedge_{b \in B} \mathcal{M}_b(f) \). First observe that using Claim 4.1.1 part (4) we get \( f \equiv \bigwedge_{b \in B} \mathcal{M}_b(f) \). In order to show \( f \equiv \bigwedge_{b \in B} \mathcal{M}_b(f) \) we show that for all \( y \not\in f \) there exists \( b \in B \) such that \( y \not\in \mathcal{M}_b(f) \), and therefore \( y \not\in \bigwedge_{b \in B} \mathcal{M}_b(f) \).

Consider \( y \in \{0, 1\}^n \) such that \( y \not\in f \) and assume, w.l.o.g., that \( C_1(y) = 0 \). Let \( b = b^1 \) be the corresponding element in \( B \), and assume, by way of contradiction that \( \mathcal{M}_b(f)(y) = 1 \). Then, by Corollary 4.1.4 there exists \( z \in \min_b(\mathcal{M}_b(f)) = \min_b(f) \) such that \( z \leq_b y \). We therefore have that \( b \leq_z z \leq_b y \). Let \( x_i \) be a variable that appears in the clause \( C_1 \). Since \( C_1(y) = C_1(b) = 0 \), we must have \( y_i = z_i = b_i \). Since this holds for all variables that appear in \( C_1 \), it implies that \( C_1(z) = 0 \) and contradicts the assumption that \( z \in f \).

The set of floor assignments of an assignment \( x \), with respect to the order relation \( b \), denoted \([x]_b\), is the set of all elements \( z <_b x \) such that there does not exist \( y \) for which \( z <_b y <_b x \) (i.e., \( z \) is strictly smaller than \( x \) relative to \( b \) and is different from \( x \) in exactly one bit).

The set of local minimal assignments of \( f \) with respect to \( b \) is:

\[
\min^*_b(f) = \{ x \mid x \in f, \text{ and } \forall y \in [x]_b, \ y \not\in f \}.
\]

The following claims bound the size of \( \min_b(f) \):

**Claim 4.1.6** Let \( f = D_1 \lor D_2 \lor \cdots \lor D_k \) be a DNF representation for \( f \). Then for every \( b \in \{0, 1\}^n \), \( |\min^*_b(f)| \leq k \).

**Proof:** Let \( D \) be one of the terms in the representation, and let \( p \) be the number of literals in \( D \). That is \( D = \bigwedge_{i=1}^{p} x^{c_i}_{i} \) (here \( x^1 = x \) and \( x^0 = \overline{x} \)). Clearly, the set \( \min_b(D) = \min^*_b(D) \) contains a single element, \( z \), defined by \( z_i = c_i \) if \( x_i \) appears in \( D \) and \( z_i = b_i \) if \( x_i \) does not appear in \( D \). Further, for any two functions \( g_1, g_2 \), \( \min^*_b(g_1 \cup g_2) \subseteq \min^*_b(g_1) \cup \min^*_b(g_2) \) and therefore \( |\min^*_b(f)| \leq |\bigcup_{i=1}^{k} \min^*_b(D_i)| \leq k \).

**Corollary 4.1.7** Let \( f = D_1 \lor D_2 \lor \cdots \lor D_k \) be a DNF representation for \( f \). Then for every \( b \in \{0, 1\}^n \), \( |\min_b(f)| \leq k \).
Proof: This follows from Claim 4.1.6, observing that by definition \( \min_b(f) \subseteq \min_b^*(f) \).

Example: Let \( f \) have the CNF representation:

\[
f = (x_1 \lor x_2 \lor x_3) \land (x_1 \lor x_2 \lor x_4) \land (\overline{x_1} \lor \overline{x_2} \lor x_3 \lor x_4)
\]

The function \( f \) has 12 (out of the 16 possible) satisfying assignments. The non-satisfying assignments of \( f \) are\(^4\): \( \{0000, 0001, 0010, 1101\} \). Using Claim 4.1.5 we get that the set \( B = \{0000, 1101\} \) is a basis for \( f \).

The sets of minimal assignments with respect to this basis are: \( \min_{0000}(f) = \{1000, 0100, 0011\} \) and \( \min_{1101}(f) = \{1100, 1111, 1001, 0101\} \). These can be easily found by drawing the corresponding lattices and checking which of the satisfying assignments of \( f \) are minimal. It is also easy to check that \( f \) can be represented as in equation (4.1) using the minimal elements identified.

### 4.2 Deduction with Models

We consider the deduction problem \( KB \models \alpha \). \( KB \) is the knowledge base, which is taken to be a propositional expression (i.e., some Boolean function), and \( \alpha \) is also a propositional expression. The assertion \( KB \models \alpha \) means that if some model \( x \in \{0, 1\}^n \) satisfies \( KB \), then it must also satisfy \( \alpha \).

Let \( \Gamma \subseteq KB \subseteq \{0, 1\}^n \) be a set of models. To decide whether \( KB \models \alpha \) we consider the straight forward model-based approach to deduction: for all the models \( z \in \Gamma \) check whether \( \alpha(z) = 1 \). If for some \( z \), \( \alpha(z) = 0 \), say “no”; otherwise say “yes”.

By definition, if \( \Gamma = KB \) this approach yields correct deduction, but representing \( KB \) by explicitly holding all the possible models of \( KB \) is not plausible. A model-based approach becomes feasible if one can make correct inferences when working with a small subset of models.

In this section we define a special collection \( \Gamma \) of characteristic models of \( KB \) and show that performing the model-based test on \( \Gamma \) yields correct deduction. We fully characterize \( \Gamma \) in terms of the Boolean function \( KB \) and the query \( \alpha \). Thus we can explore the trade-off between the logical and a model-based approach in terms of size of representation as well as efficiency of reasoning.

#### 4.2.1 Exact Deduction

**Definition 4.2.1** Let \( \mathcal{F} \) be a class of functions, and let \( B \) be a basis for \( \mathcal{F} \). For a knowledge base \( KB \in \mathcal{F} \) we define the set \( \Gamma = \Gamma_{KB}^B \) of characteristic models to be

\[^4\]An element of \( \{0, 1\}^n \) denotes an assignment to the variables \( x_1, \ldots, x_n \) (i.e., 0011 means \( x_1 = x_2 = 0 \), and \( x_3 = x_4 = 1 \)).
the set of all minimal assignments of KB with respect to the basis B. Formally,
\[ \Gamma_{KB}^B = \bigcup_{b \in B} \{ z \in \min_b(KB) \} \].

**Theorem 4.2.1** Let KB, \( \alpha \in \mathcal{F} \) and let B be a basis for \( \mathcal{F} \). Then KB \( \models \alpha \) if and only if for every \( u \in \Gamma_{KB}^B \), \( \alpha(u) = 1 \).

**Proof:** Clearly, \( \Gamma = \Gamma_{KB}^B \subseteq KB \) and therefore, if there exists \( z \in \Gamma \) such that \( \alpha(z) = 0 \) then KB \( \not\models \alpha \). For the other direction assume that for all \( u \in \Gamma \), \( \alpha(u) = 1 \).

We will show that if \( y \in KB \), then \( \alpha(y) = 1 \). From Corollary 4.1.4, since B is a basis for \( \alpha \) and for all \( u \in \Gamma \), \( \alpha(u) = 1 \), we have that
\[ \forall u \in \Gamma, \ \forall b \in B, \ \exists v_{u,b} \in \min_b(\alpha) \text{ such that } u \geq_b v_{u,b}. \] (4.2)

Consider now a model \( y \in KB \). Again, Corollary 4.1.4 implies that
\[ \forall b \in B, \ \exists z \in \min_b(KB) \text{ such that } y \geq_b z. \] (4.3)

By the assumption, since \( \min_b(KB) \subseteq \Gamma \), all the elements \( z \) identified in Equation 4.3 satisfy \( \alpha \) and therefore, as in Equation 4.2 we have that
\[ \forall z \in \min_b(KB), \ \exists v_{z,b} \in \min_b(\alpha) \text{ such that } z \geq_b v_{z,b}. \] (4.4)

Substituting Equation 4.4 into Equation 4.3 gives the required condition on \( y \in KB \):
\[ \forall b \in B, \ \exists v_{(z),b} \in \min_b(\alpha) \text{ such that } y \geq_b v_{(z),b} \]
which implies, by Corollary 4.1.4, that \( \alpha(y) = 1 \).

The above theorem requires that KB and \( \alpha \) can be described by the same basis B. This requirement is somewhat relaxed in the following theorem.

**Theorem 4.2.2** Let KB be a propositional theory with basis B and let \( \alpha \) be a query with basis \( B' \). Then \( \Gamma_{KB}^{B \cup B'} \) is a model-based representation for the inference problem KB \( \models \alpha \). That is, KB \( \models \alpha \) if and only if for every \( u \in \Gamma_{KB}^{B \cup B'} \), \( \alpha(u) = 1 \).

**Proof:** It is clear, from Eq. 4.1 and Claim 4.1.1 part (4) that \( B \cup B' \) is a basis both for KB and \( \alpha \). Therefore, Theorem 4.2.1 implies the result.

**Example:** (continued) The set \( \Gamma \) relative to \( B = \{0000, 1101\} \) is:
\[ \Gamma = \{1000, 0100, 0011, 1100, 1111, 1001, 0101\} \]. Note that it includes only 7 out of the 12 satisfying assignments of \( f \). Since model-based deduction does not make mistakes on queries that are implied by \( f \) we concentrate in our examples on queries that are not implied by \( f \).

To exemplify Theorem 4.2.1 consider the query \( \alpha_1 = \overline{x_2} \land \overline{x_3} \rightarrow x_4 \). This is equivalent to \( x_2 \lor x_3 \lor x_4 \) which is falsified by 0000 so \( B \) is a basis for \( \alpha \). Reasoning with \( \Gamma \) will find the counterexample 1000 and will therefore conclude \( f \not\models \alpha_1 \).
The query $\alpha_2 = x_1 \land x_3 \rightarrow x_2$ is equivalent to $\overline{x_1} \lor x_2 \lor \overline{x_3}$ which is not falsified by the basis $B$. Therefore $B$ is not a basis for $\alpha_2$ and model-based deduction might be wrong. Indeed reasoning with $\Gamma$ will not find a counterexample and will conclude $f \models \alpha_2$ (it is wrong since the assignments 1010, 1011 satisfy $f$ but not $\alpha_2$).

Next, to exemplify Theorem 4.2.2 consider adding a basis element for $\alpha_2$. This could be either 1010, or 1011. Choosing 1010, the set of additional minimal elements in $\Gamma$ is $\{1010\}$, and reasoning with $\Gamma$ would be correct on $\alpha_2$.

### 4.2.2 Exact Deduction with Approximate Theories

We have shown how to perform deduction with the set of characteristic models $\Gamma^B_{KB}$, where $B$ is a basis for the knowledge base $KB$ and the class $\mathcal{G}$ of possible queries. In this section we show that if $B$ is a basis for the class $\mathcal{G}$ of queries, we can answer those queries even when $B$ is not a basis for the knowledge base $KB$.

In order to do that we use the notion of the least upper bound (LUB) (Selman and Kautz, 1991) of $KB$ in the class of functions with basis $B$, which we have already introduced and discussed in Section 2.3.3.

To facilitate the presentation we first recall the definition of a theory approximation and prove an important property of this notion. We then prove a representation theorem for the theory approximation: we show how to represent the LUB of $KB$ with respect to $B$ using the characteristic models of $KB$ with respect to $B$. Finally, we show the implication to reasoning, and in particular, to exact deduction with common queries.

**Definition 4.2.2 (Least Upper-bound)** Let $\mathcal{F}, \mathcal{G}$ be families of propositional languages. Given $f \in \mathcal{F}$ we say that $f_{\text{lub}} \in \mathcal{G}$ is a $\mathcal{G}$-least upper bound of $f$ if and only if $f \subseteq f_{\text{lub}}$ and there is no $f' \in \mathcal{G}$ such that $f \subseteq f' \subseteq f_{\text{lub}}$.

These bounds are called $\mathcal{G}$-approximations of the original theory $f$. The significance of the LUB notion to this discussion is due to the following theorem:

**Theorem 4.2.3** Let $\mathcal{G}$ be a class of functions with basis $B$, and $\alpha \in \mathcal{G}$. The deduction problem $KB \models \alpha$ is equivalent to the deduction problem $KB_{\text{lub}} \models \alpha$ (where the least upper bound is taken with respect to $\mathcal{G}$).

**Proof:** Assume that $KB \models \alpha$. Since $\alpha \in \mathcal{G}$, by the definition of the least upper bound, $KB_{\text{lub}} \models \alpha$. For the other direction, since by definition $KB \models KB_{\text{lub}}$, $KB_{\text{lub}} \models \alpha$ implies that $KB \models \alpha$. \hfill $\square$

The next theorem characterizes the $\mathcal{G}$-LUB of a function and shows that it is unique.

**Theorem 4.2.4** Let $f$ be any propositional theory and $\mathcal{G}$ a class of all propositional theories with basis $B$. Then

$$f_{\text{lub}} = \bigwedge_{b \in B} \mathcal{M}_b(f).$$
Proof: Define $g = \bigwedge_{b \in B} M_b(f)$. We need to prove that (1) $f \subseteq g$, (2) $g \in \mathcal{G}$ and (3) there is no $f' \in \mathcal{G}$ such that $f \subset f' \subset g$. (1) is immediate from Claim 4.1.1 part (4). To prove (2) we need to show that $B$ is a basis for $g$. Indeed,

$$
\bigwedge_{b \in B} M_b(g) = \bigwedge_{b \in B} M_b(\bigwedge_{b \in B} M_b(f)) \subseteq (\bigwedge_{b \in B} M_b(f)) \bigwedge_{b, b \neq b_j} M_{b_i} M_{b_j}(f) \\
= g \bigwedge_{b, b \neq b_j} M_{b_i} M_{b_j}(f) \\
\subseteq g.
$$

Using Claim 4.1.1 part (4) again we get $g \subseteq \bigwedge_{b \in B} M_b(g)$ and therefore $\bigwedge_{b \in B} M_b(g) = g$, that is $g \in \mathcal{G}$. Finally, to prove (3) assume that there exists $f' \in \mathcal{G}$ such that $f \subseteq f'$. Then,

$$
g = \bigwedge_{b \in B} M_b(f) \subseteq \bigwedge_{b \in B} M_b(f') = f',
$$

where the last equality results from the fact that $f' \in \mathcal{G}$. Therefore, $g = f_{\mathit{ub}}$. ■

The following theorem can be seen as a generalization of Theorem 4.2.1, in which we do not require that the basis $B$ is the basis of $KB$.

**Theorem 4.2.5** Let $KB \in \mathcal{F}$, $\alpha \in \mathcal{G}$ and let $B$ be a basis for $\mathcal{G}$. Then $KB \models \alpha$ if and only if for every $u \in \Gamma_{KB}^B$, $\alpha(u) = 1$.

Proof: We have shown in Theorem 4.2.4 that

$$
KB_{\mathit{ub}} = \bigwedge_{b \in B} M_b(KB) = \bigwedge_{b \in B} \bigvee_{z \in \min_b(KB)} M_b(z).
$$

By Theorem 4.2.1, since $\alpha(u) = 1$ for every $u \in \Gamma_{KB}^B$, we have that $KB_{\mathit{ub}} \models \alpha$ and therefore $KB \models \alpha$. On the other hand, since $\Gamma_{KB}^B \subseteq KB$, if for some $u \in \Gamma_{KB}^B$, $\alpha(u) = 0$, $KB \not\models \alpha$. ■

A result similar to the corollary that follows, for the case in which $\mathcal{G}$ is the class of Horn theories, is discussed in (Kautz and Selman, 1991; Cadoli, 1993).

**Corollary 4.2.6** Model-based Reasoning with the least upper bound (with respect to the language $\mathcal{G}$) of a theory $KB$ is correct for all queries in $\mathcal{G}$.

**Example:** (continued) The Horn basis for our example is: $B_H = \{1111, 1110, 1101, 1011, 0111\}$ (see Claim 4.3.1). The minimal elements with respect to 1101 were given before. Each of the models 1111, 0111, 1011, 1110 satisfies $f$ and therefore for each of these, $\min_b(f) = b$ and together we get that $\Gamma_{f}^B = \{1111, 0111, 1011, 1100, 1001, 0101, 1110\}$.

For the query $\alpha_3 = x_1 \land x_3 \rightarrow (x_2 \lor x_4)$, which is not Horn, reasoning with $\Gamma_{f}^B$ will be wrong. For the Horn query $\alpha_2 = x_1 \land x_3 \rightarrow x_2$, reasoning with $\Gamma_{f}^B$ will find the counterexample 1011 and therefore be correct.
We finish this section on deduction by considering the question of the necessity of the complete set of characteristic models for correct deduction. In Theorems 4.2.1 and 4.2.5 we have proved that a complete set of characteristic models is sufficient to support deduction. Are all of them necessary? We answer this question affirmatively in the following theorem:

**Theorem 4.2.7** Let $B \subseteq \{0, 1\}^n$ be a set of assignments, and let $\mathcal{G}$ be the class of all Boolean functions that can be represented using $B$ as a basis. Let $KB \in \mathcal{F}$ and $\alpha \in \mathcal{G}$. Assume that $R \subseteq KB$ satisfies the property that $KB \models \alpha$ if and only if for every $u \in R$, $\alpha(u) = 1$. Then $\Gamma_{KB}^B \subseteq R$.

**Proof:** Suppose there exists a set $R$ that satisfies the above property and such that $\Gamma_{KB}^B \not\subseteq R$, and consider $x \in \Gamma_{KB}^B \setminus R$. We show that there is a function $\alpha' \in \mathcal{G}$ such that for all $u \in R$, $\alpha'(u) = 1$, but still $KB \not\models \alpha'$, yielding a contradiction.

Indeed, define $\alpha' = R_{\text{LUB}}$. (That is, the LUB with respect to $B$ of the function whose satisfying assignments are exactly the elements of $R$.) Then, by definition, $\alpha' \in \mathcal{G}$, and $R \subseteq \alpha'$, that is, all the elements in $R$ satisfy $\alpha'$. However, $KB \not\models \alpha'$: to see that, notice that since $x \in \Gamma_{KB}^B$, $x$ is a minimal model with respect to some $b \in B$. With respect to this element $b$, we get that for all $z \in KB$ and, in particular, for all $z \in R$, $x \not\leq_b z$, that is $x \not\in \mathcal{M}_b(R)$. Using Theorem 4.2.4 we get that $x \not\in \alpha'$.

Note the difference in the premises of Theorem 4.2.7 and the previous two theorems, Theorem 4.2.1 and 4.2.5. Theorem 4.2.7 shows that every element of the set $\Gamma$ is necessary in order to get correct deduction. What the proof shows is that there exists a function $\alpha'$ in the class represented by $B$, which necessitates the use of each element $x$ in $\Gamma$. Note that, in general, if $B$ is a basis for $\mathcal{G}$ it does not mean that all functions in the class represented by $B$ are in the class $\mathcal{G}$, and therefore the premises of the previous theorems are not enough to yield this result. (We discuss this point further in Section 4.5.1.)

In the next section we discuss with some details the basis $B_H$ of the class of Horn theories. We note that in this case (as well as in the case of the basis $B_{kq}$ of $k$-quasi-Horn functions), the bases represent those classes exactly. That is, a function is $k$-quasi-Horn if and only if it can be represented using $B_{kq}$.

### 4.3 Horn Theories

We consider in detail the case of Horn formulas and show that in this case our notion of characteristic models coincides with the notion introduced in (Kautz, Kearns, and Selman, 1993; Beeri et al., 1984). We further discuss the issue of answering all CNF queries. In (Kautz, Kearns, and Selman, 1993) the deduction theorem was extended to answer any such query. This extension relies on a special property of Horn formulas and does not hold in the general case. We give an example that explains this phenomenon. We start by showing that Horn formulas have a small basis.
Claim 4.3.1 The set \( B_H = \{ u \in \{0,1\}^n \mid \text{weight}(u) \geq n - 1 \} \) is a basis for any Horn CNF function.

Proof: Let \( KB \) be any Horn function. Denote by \( b^{(0)} \) the basis element with weight \( n \) and by \( b^{(i)} \) the basis element with the \( i \)th bit set to zero and all the others set to one. By Claim 4.1.5 it is enough to show that if \( C \) is a clause in the CNF representation of \( KB \) then it is falsified by one of the basis elements in \( B \). Indeed, if \( C \) is a clause in which all the literals are negative, then it is falsified by \( b^{(0)} \). If \( x_k \) is the only variable that appears un-negated in \( C \) then \( C \) is falsified by \( b^{(k)} \).

4.3.1 Characteristic Models

In order to relate to the results from (Kautz, Kearns, and Selman, 1993) we need a few definitions presented there.

For \( u, v \in \{0,1\}^n \), we define the intersection of \( u \) and \( v \) to be the assignment \( z \in \{0,1\}^n \) such that \( z_i = 1 \) if and only if \( u_i = 1 \) and \( v_i = 1 \) (i.e., the bitwise logical-and of \( u \) and \( v \)).

The closure of \( S \subseteq \{0,1\}^n \), denoted \( \text{closure}(S) \), is defined as the smallest set containing \( S \) that is closed under intersection.

Let \( KB \) be a Horn theory. The set of the Horn characteristic models of \( KB \), denoted here \( \text{char}_H(KB) \) is defined as the set of models of \( KB \) that are not the intersection of other models of \( KB \). Formally,

\[
\text{char}_H(KB) = \{ u \in KB \mid u \notin \text{closure}(KB \setminus \{u\}) \}. \quad (4.5)
\]

The following claim should be attributed to McKinsey (1943) (note that the definitions there are different; an adaptation of this proof to the propositional terminology can be found in (Khardon and Roth, 1994d)). A different proof of this property is given by Dechter and Pearl (1992).

Claim 4.3.2 (McKinsey (1943)) A theory is Horn if and only if its set of models is closed under intersection.

Based on this characterization of Horn theories, it is clear that if \( KB \) is a Horn theory and \( M \subseteq KB \) any subset of models, then \( \text{closure}(M) \subseteq \text{closure}(KB) = KB \). In (Kautz, Kearns, and Selman, 1993) it is shown that if we take \( M = \text{char}_H(KB) \), then we get

\[
\text{closure}(\text{char}_H(KB)) = \text{closure}(KB) = KB.
\]

In particular, Equation 4.5 implies that \( \text{char}_H(KB) \) is the smallest subset of \( KB \) with that property. Based on this it is then shown that model-based deduction using \( \text{char}_H(KB) \) yields correct deduction. In the following we show that with respect to the basis \( B_H \) from Claim 4.3.1, and for any Horn theory \( KB \), \( \text{char}_H(KB) = \Gamma_B^{KB} \). Therefore \( \text{char}_H(KB) \) is an instance of the theory developed in Section 4.2, and we can reason with it according to Theorem 4.2.1.
Theorem 4.3.3 Let $KB$ be a Horn theory and $B_H = \{u \in \{0,1\}^n \mid \text{weight}(u) \geq n-1\}$. Then, $\text{char}_H(KB) = \Gamma_{KB}^{B_H}$.

Proof: Denote $\Gamma = \Gamma_{KB}^{B_H}$. In order to show that $\text{char}_H(KB) \subseteq \Gamma$, it is sufficient to prove that $KB = \text{closure}(\Gamma)$. This is true since $\text{char}_H(KB)$ is the smallest subset of $KB$ with that property.

Consider $x \in KB$, Corollary 4.1.4 implies that for all $b^{(i)} \in B$, there exists $u^{(i)} \in \min_{b^{(i)}}(f)$ such that $x \geq_{b^{(i)}} u^{(i)}$. We claim that

$$x = \land_{\{k:x_k=0\}} u^{(k)} \in \text{closure}(\Gamma).$$

To see that, consider first the zero bits of $x$. Let $x_j = 0$, this implies that $u^{(j)}$ is in the intersection and that it satisfies $x \geq_{b^{(j)}} u^{(j)}$. Since $x_j = 0$ and $b^{(j)}_j = 0$ the fact $x_j \geq_{b^{(j)}} u^{(j)}_j$ implies $u^{(j)}_j = 0$, and the intersection on this bit is also 0.

Consider now the case $x_j = 1$. Since all the $u^{(k)}$ in the intersection are such that $x_k = 0$, the order relation on the $j$th bit is always the reversed order, $\leq_1$. That is, all the $u^{(k)}$ in the intersection satisfy $1 = u^{(j)}_j \geq_1 u^{(k)}_j$. This implies that for all the $u^{(k)}$ in the intersection $u^{(j)}_j = 1$ and the intersection on this bit is also 1. This completes the proof of $\text{char}_H(KB) \subseteq \Gamma$.

To prove $\Gamma \subseteq \text{char}_H(KB)$, we show that if $x \in \Gamma$, $x$ cannot be represented as $x = y \land z$ where $y, z \in KB$ and $x \neq y, z$. Since $\text{char}_H(KB)$ is the collection of all those elements in $KB$ (from Equation 4.5), we get the result.

Consider $x \in \min_{b^{(k)}}(KB) \subseteq \Gamma$, and suppose by way of contradiction that $\exists y, z \in KB$ such that $x = y \land z$ and $x \neq y, z$. Fix the order relation $b^{(k)}$ and consider the indices of $x$. First consider an index $i \neq k$. Since $b^{(k)}_i = 1$ the order relation of the $i$th index is the reversed one. Now, if $y_i = z_i$ then $x_i = y_i = z_i$, and if $y_i \neq z_i$ then $x_i = 0$. Therefore, in both cases we get that for all $i \neq k$ $x_i \geq_{b^{(k)}} y_i$ and $x_i \geq_{b^{(k)}} z_i$. For the case $k=0$, the indices $i \neq k$ include all the bits. This implies $x \geq_{b^{(k)}} y$ and $x \geq_{b^{(k)}} z$ and since $x \in \min_{b^{(k)}}(KB)$, this contradicts the assumption that $x \neq y, z$, and therefore proves the claim.

Otherwise, when $k \neq 0$ we consider also the order relation of the $k$th index, which is the usual order. Again, if $y_k = z_k$ then $x_k = y_k = z_k$ and if $y_k \neq z_k$ then $x_k = 0$. This implies that $x_k \geq_{b^{(k)}} y_k$ or $x_k \geq_{b^{(k)}} z_k$.

Together with the case $i \neq k$ we get that $x \geq_{b^{(k)}} y$ or $x \geq_{b^{(k)}} z$ (depends on whether $z_k = 0$ or $y_k = 0$). But since $x \in \min_{b^{(k)}}(KB)$, this contradicts the assumption that $x \neq y, z$, and completes the proof. \[\blacksquare\]

4.3.2 General Queries

Kautz, Kearns, and Selman (1993) show that in case of Horn theories one can answer any CNF query without re-computing the characteristic models. While we have shown that our model-based representation coincides with that of (Kautz, Kearns, and Selman, 1993) it turns out that the ability to answer any query relies on a
special characteristic of Horn theories, and does not generalize to other propositional theories. We next give a counterexample that exemplifies this. The deduction scheme in (Kautz, Kearns, and Selman, 1993) when $\alpha$ is a general CNF expression, utilizes the following theorem$^5$.

**Theorem 4.3.4** Let $KB$ be a Horn theory and $\alpha$ any disjunction. If $KB \models \alpha$ then there exists a Horn disjunction $\beta$ such that $KB \models \beta$ and $\beta \models \alpha$.

Together with the following observations:

1. Every disjunction $\alpha$ can be represented as $\alpha = \beta_1 \lor \ldots \lor \beta_k$, where the $\beta_i$ are Horn disjunctions.

2. $KB \models \alpha_1 \alpha_2$ if and only if $KB \models \alpha_1$ and $KB \models \alpha_2$.

Observation (2) implies that it is enough to consider queries that are disjunctions. Given $\alpha$, the deduction scheme in (Kautz, Kearns, and Selman, 1993) decomposes it into the Horn disjunction $\beta$s and tests deduction against the $\beta$s. By Theorem 4.3.4, at least one of the $\beta$s is implied by $KB$. While the observations above are true even for non-Horn theories, Theorem 4.3.4 depends on $KB$ being Horn, as the following example shows.

**Example** [Theorem 4.3.4 does not hold for non-Horn languages]

Let:

$$KB = (x_1 \lor x_2 \lor \overline{x_3} \lor \overline{x_7}) \land (x_3 \lor x_5 \lor \overline{x_6}).$$

$$\alpha = x_1 \lor x_2 \lor \overline{x_4} \lor x_5 \lor \overline{x_6}$$

The knowledge base $KB$ is not a Horn theory, and it is easy to check that $KB \not\models \alpha$. However, there is no disjunction $\beta$ such that $KB \models \beta \models \alpha$.

### 4.4 The Size of Model Based Representations

The complexity of model-based reasoning is directly related to the number of models in the representation. It is therefore important to compare this size with the size of other representations of the same function. In the previous section we have shown that our model-based representation is the same as that of (Kautz, Kearns, and Selman, 1993) when the theory is Horn. In (Kautz, Kearns, and Selman, 1993) examples are given for large Horn theories with a small set of characteristic models and vice versa, but it was not yet understood when and why it happens. Our results imply that the set of characteristic models of a Horn theory is small if the size of a DNF description for the same theory is small. The other direction is however not true (i.e., there are Horn theories with a small set of characteristic models but an exponential size DNF). We start with a bound on the size of the model-based representation.

---

$^5$This theorem follows from McKinsey’s proof of Claim 4.3.2. In (Kautz, Kearns, and Selman, 1993) it is derived, in a different way, using a completeness theorem for resolution given in (Slage, Chang, and Lee, 1969) (see also (Khardon and Roth, 1994d) for a discussion).
Lemma 4.4.1 Let \( B \) be a basis for the knowledge base \( KB \), and denote by \(|DNF(KB)|\) the size of its DNF representation. Then, the size of the model-based representation of a knowledge base \( KB \) is

\[
|\Gamma^B_{KB}| \leq \sum_{b \in B} |\text{min}_b(KB)| \leq |B| \cdot |DNF(KB)|.
\]

Proof: The lemma follows from Corollary 4.1.7.

As the following claim shows, this bound is actually achieved for some functions. For the next claim we would need the following terminology. A term \( t \) is an implicant of a function \( f \), if \( t \models f \). A term \( t \) is a prime implicant of a function \( f \), if \( t \) is an implicant of \( f \) and the conjunction of any proper subset of the literals in \( t \) is not an implicant of \( f \).

Claim 4.4.2 For any \( b \)-monotone function \( f \), \(|\text{min}_b(f)| = |DNF(f)|\).

Proof: We first consider monotone functions (i.e. \( 0^n\)-monotone). It is well known that for a monotone function there is a unique DNF representation in which each term is a prime implicant. Let \( f \) be a monotone function and consider this representation for \( f \). Similar to Claim 4.1.6 we can map every term in the representation to its corresponding minimal element. Moreover, since the terms are monotone and the order relation is \( 0^n \), each of these minimal elements is indeed a minimal element of \( f \) (otherwise one of the terms in the representation is not a prime implicant). So there is a one to one correspondence between prime implicants and minimal assignments of \( f \) with respect to \( b = 0^n \), and \(|\text{min}_b(f)| = |DNF(f)|\). The same arguments hold for any \( b \)-monotone function with respect to the order relation \( b \) (one can simply rename the variables) and therefore \(|\text{min}_b(f)| = |DNF(f)|\).

Claim 4.4.2 explains the two examples in (Kautz, Kearns, and Selman, 1993). Both examples are \( 1^n\)-monotone Horn functions, one has a small DNF and the other has an exponentially large DNF. We note that exponential size model-based representations are not restricted to happen in \( b \)-monotone functions. One can easily construct such functions by using, for example, a conjunction of several functions each \( b \)-monotone with a different \( b \) (of course the DNF size has to be exponential here too). The following claim shows that DNF size is not always needed. There are theories for which the DNF size is exponential but the size of the model-based representation, and therefore also model-based reasoning is polynomial.

Claim 4.4.3 There exist Horn formulas with an exponential size DNF and a set \( \Gamma^B_{d_H} \) of linear size.

Proof: For each \( n \) we exhibit a formula \( f \) with the required property. The function

\[
f = (\overline{x_1} \lor \overline{x_2} \cdots \lor \overline{x_{\sqrt{n}-1}} \lor x_{\sqrt{n}}) \land \cdots \land (\overline{x_{n-\sqrt{n}+1}} \lor \overline{x_{n-\sqrt{n}+2}} \lor \cdots \lor \overline{x_{n-1}} \lor x_n)
\]
is clearly in Horn form.

The size of its DNF representation is $\sqrt{n^n}$. This is easy to observe by renaming the negative literals as its negation. This yields a monotone formula in which each term we get, by multiplying one variable from each clause, is a prime implicant.

The set $\Gamma$ is of size $< 2n$. Recall that $b^{(i)} \in B_H$ denotes the basis element in which the variable $x_i$ is assigned 0, and $b^{(0)} = 1^n$. First observe that for $i \neq k\sqrt{n}$, $b^{(i)}$ is a satisfying assignment of $f$ and therefore has only one minimal element (that is, itself). For $i = k\sqrt{n}$, $b^{(i)}$ is not a satisfying assignment of $f$. There is however only one clause, $C_i$, not satisfied by $b^{(i)}$, the clause which includes the variable $x_i$. Now, since each variable appears only once in $f$, each of the variables in $C_i$ we flip yields a satisfying assignment which is minimal. This contributes $\sqrt{n}$ minimal assignments. (Flipping variables not in $C_i$ does not contribute minimal assignments with respect to $b^{(i)}$.) One last note is that each of these $b^{(i)}$ would have $1^n$ as one of the minimal assignments, so we need to count it only once, and count $(\sqrt{n} - 1)$ for each of the $b^{(i)}$'s.

Altogether there are $(n - \sqrt{n}) + \sqrt{n}(\sqrt{n} - 1) + 1 = 2(n - \sqrt{n}) + 1$ minimal elements.

Considering model-based representations, Claim 4.4.2 implies that for every basis there is a function which has an exponential number of characteristic models. Nevertheless, one might hope that there is a basis for which least upper bounds will always have small representations in some (maybe other) form that admits fast reasoning. Kautz and Selman (Kautz and Selman, 1992) show that for Horn representations this is not the case. In particular they show that unless NP $\subseteq$ non-uniform P there exists a function whose Horn LUB does not have a short representation that allows for efficient reasoning. This can be generalized, using essentially the same proof, to hold for every fixed basis and in particular $k$-quasi-Horn, $\log n$ CNF, and monotone functions. We therefore have the following theorem:

**Theorem 4.4.4** Unless NP $\subseteq$ non-uniform P, for every fixed basis $B$ there exists a function whose LUB does not have a short representation which admits efficient reasoning.

### 4.5 Applications

In Section 4.2 we developed the general theory for model based deduction. In this section we discuss applications of this theory to specific propositional languages and to the problem of structure identification.

Our basic result (Theorem 4.2.1) assumed that the knowledge base and the query share the same basis. We give such queries a special status.

---

\(^6\)This issue has been brought to our attention by Henry Kautz and Bart Selman.
Definition 4.5.1 Let $B$ be a basis for $KB$. A query $\alpha$ is relevant to $KB$ if $B$ is a basis for $\alpha$.

Theorem 4.2.2 suggests a way in which one can overcome the difficulty in the case where the basis $B$ of $KB$ is not a basis for the query $\alpha$. This can be done by: (1) adding the basis $B'$ of the query to the knowledge base basis, and (2) computing additional characteristic models based on the new basis.

Claim 4.1.5 suggests a simple way for computing the basis for a given query, as required in (1). The problem of computing additional characteristic models, however, is in general a hard problem that we do not address here. (See, (Kavvadias, Papadimitriou, and Siedri, 1993)).

Neither do we consider computing additional models in an on-line process performed for each query. At this point we assume that the knowledge base is given in the form of its set of characteristic models. In the next chapter, however, we embed this model-based representation within the Learning to Reason framework and by that side-step the hardness of computing the characteristic models.

We say that queries are common if they are taken from some common propositional language as defined below.

Definition 4.5.2 A language $\mathcal{F}$ is common if there is a small (polynomial size) fixed basis for all $f \in \mathcal{F}$. The set $\mathcal{L}_C$ is the set of common languages.

Important examples of common languages are: (1) Horn-CNF formulas, (2) reversed Horn-CNF formulas (CNF with clauses containing at most one negative literal), (3) $k$-quasi-Horn formulas (a generalization of Horn theories in which there are at most $k$ positive literals in each clause), (4) $k$-quasi-reversed-Horn formulas and (5) log $n$CNF formulas (CNF in which the clauses contain at most $O(\log n)$ literals). Any formula that can be represented as a CNF with clauses from any combination of the above is also in $\mathcal{L}_C$. The fixed bases for these languages are discussed in the following subsection.

4.5.1 Languages with a Small Basis

In Claim 4.3.1 we have shown that Horn formulas have a short basis. A similar construction yields a basis for reversed Horn, $k$-quasi-Horn formulas, and $k$-quasi-reversed-Horn formulas.

Claim 4.5.1 There is a polynomial size basis for: reversed Horn formulas, $k$-quasi-Horn formulas, and $k$-quasi-reversed-Horn formulas.

Proof: The analysis is very similar to the one in Claim 4.3.1. By flipping the polarity of all bits in $B_H$ we can get a basis for reversed Horn. Similarly, using

---

7Note that a fixed basis uniquely characterizes a family of Boolean functions which can be represented using it. There are of course other ways to characterize classes of functions which do not correspond to any basis (e.g. some subset of DNF).
the set $B_{H_k} = \{ u \in \{0,1\}^n \mid \text{weight}(u) \geq n - k \}$ we get a basis for $k$-quasi-Horn, and flipping the polarity of all bits in $B_{H_k}$ we get a basis for $k$-quasi-reversed-Horn formulas.

We next consider the expressive class of $\log n$ CNF formulas, in which there are up to $O(\log n)$ variables in a clause, and show that it has a polynomial size basis.

An $(n, k)$-universal set is a set of assignments $\{d_1, \ldots, d_l\} \subseteq \{0,1\}^n$ such that every subset of $k$ variables assumes all of its $2^k$ possible assignments in the $d_i$'s. It is known (Alon et al., 1992) that for $k = \log n$ one can construct $(n, k)$-universal sets of polynomial size.

Claim 4.5.2 (Bshouty (1993a)) Let $B$ be an $(n, k)$-universal set. Then $B$ is a basis for any $k$CNF $KB$.

Proof: By Claim 4.1.5 it is enough to show that if $C$ is a clause in the $k$CNF representation of $KB$ then it is falsified by one of the basis elements in $B$. Let $C = l_i \lor \ldots \lor l_k$ be a clause in the CNF representation of $KB$, where $l_i \in \{x_i, \overline{x_i}\}$. Let $a \in \{0,1\}^n$ be an assignment. Then the value $C(a)$ is determined only by $a_i, \ldots, a_k$ and since $B$ is an $(n, k)$-universal set, there must be an element $b \in B$ for which $C(b) = 0$.

We note that in general the fact that $B$ is a basis for the class of functions $\mathcal{F}$ does not mean that all functions with basis $B$ are in $\mathcal{F}$. For example, given a particular $(n, \log n)$ set $B$, many other Boolean functions, outside of $\log n$CNF, have $B$ as their basis. Thus, $f_{lub}$ with respect to $B$, an $(n, \log n)$ set, is not equivalent to the least upper bound in the class $\log n$CNF but rather it is the least upper bound in the richer class of functions with basis $B$. It is easy to observe that this does not happen when using the basis $B_H$ of the class of Horn theories, as well as in the case of the basis $B_{H_k}$ of $k$-quasi-Horn functions. In this case, the class of Boolean function with basis $B_H (B_{H_k})$ is exactly the class of Horn theories ($k$-quasi-Horn functions).

4.5.2 Main Applications

In the case of common or relevant queries, reasoning involves the evaluation of a propositional formula on a polynomial number of assignments. This is a very simple and easily parallelizable procedure. Moreover, Theorem 4.2.5 shows that in order to reason with common queries, we need not use the basis of $KB$ at all, and it is enough to represent $KB$ by the set of characteristic models with respect to the basis of the query language. Lemma 4.4.1 together with Theorems 4.2.1,4.2.2,4.2.4 and 4.2.5 imply the following general applications of our theory:

Theorem 4.5.3 Any function $f : \{0,1\}^n \rightarrow \{0,1\}$ that has a polynomial size representation in both DNF and CNF form can be described with a polynomial size set of characteristic models.
Theorem 4.5.4 Any function \( f : \{0, 1\}^n \rightarrow \{0, 1\} \) in any language \( \mathcal{F} \in \mathcal{L}_C \) which has a polynomial size DNF representation can be described with a polynomial size set of characteristic models.

Theorem 4.5.5 Let \( KB \) be a knowledge base (on \( n \) variables) that can be described with a polynomial size set \( \Gamma \) of characteristic models. Then, for any relevant or common query, model-based deduction using \( \Gamma \), is both correct and efficient.

Theorem 4.5.6 Let \( KB \) be a knowledge base (on \( n \) variables) that can be described with a polynomial size DNF. Then there exists a fixed, polynomial size set of models \( \Gamma \), such that for any common query, a model-based deduction using \( \Gamma \), is both correct and efficient.

The results in this chapter are concerned mainly with propositional languages. While many AI formalizations use first order logic as its main tool, some applications do not need the full power of first order logic. It is quite easy to observe that any such formalization which is function free and has a finite number of constants can be mapped into a finite propositional language. Furthermore, a function free universally quantified sentence in Horn form (or any other language with a fixed basis) remains in Horn form\(^8\) in the new propositional domain. These observations imply that our results hold for these restricted first order logic formalizations, where the polynomial bounds are relative to the number of variables in the propositional domain.

4.5.3 Structure Identification

We briefly describe another application of model based reasoning.

Dechter and Pearl investigate in (Dechter and Pearl, 1992) the problem of identifying tractable classes of CNF formulas. In particular they consider the following problem: Given a set \( \rho \) of models, can one:

1. Find a Horn theory \( f \) such that \( f = \rho \), if one exists.

2. Find a Horn theory \( f \) such that \( \rho \subseteq f \) and there is no Horn function \( g \) such that \( \rho \subseteq g \subseteq f \).

In (Dechter and Pearl, 1992) it is shown (“Horn theories are identifiable”, Corollary 4.11) that when \( \rho \) is a set of models of a Horn theory, a theory \( f \) can be found in time polynomial in \( |\rho| \). The second problem (“strong-identifiability of Horn-theories”), which is just the problem of finding \( \rho^H_{\text{lub}} \) (i.e., the least upper bound of \( \rho \) with respect to Horn) is left open.

\(^8\)We note that the CNF formula size grows exponentially with the number of quantifiers. This size however does not effect our results as we are interested in the size of the basis and the size of the DNF formula.
The interest in the question of the identifiability and strong identifiability of Horn theories is motivated by the fact that Horn theories are tractable, and in particular, given a CNF formula in a Horn form, reasoning with it can be performed efficiently. Therefore, the significance of the strong-identifiability is that given a set \( \rho \) of models, one could perform efficient reasoning with a theory that is the best Horn approximation of \( \rho \). We show now that the techniques developed in this chapter solve that above problem (without having to find a Horn formula).

**Claim 4.5.7** Let \( \rho \) be a set of models and \( g \) the least Horn upper bound of \( \rho \). Then,

(i) A closed form formula for the required approximation can be written as a conjunction of DNF formulas:

\[
 g = \bigwedge_{B_H} \bigvee_{z \in \rho} M_l z. \tag{4.6}
\]

(ii) For every query \( \alpha \), the reasoning problem \( g \models \alpha \) can be answered correctly using the set \( \rho \).

**Proof:** (i) is immediate from Theorem 4-2-4, since \( g = \rho^{H \text{ub}} \). For (ii) notice that if the query \( \alpha \) is a Horn query, then model based reasoning with \( \rho \) is correct, by Theorem 4-2-5. For correct reasoning with general queries, it is possible to use the reasoning algorithm from (Kautz, Kearns, and Selman, 1993).

### 4.6 Abduction with Models

We consider in this section the question of performing abduction using a model-based representation. In (Kautz, Kearns, and Selman, 1993) it is shown that for a Horn theory \( KB \), abduction can be done in polynomial time using characteristic models, although using formula based representation the problem is NP-Hard (Selman, 1990). In this section we show that if we add a few base assignments to our basis, the algorithm presented there works in the general case too.

Abduction is the task of finding a minimal explanation to some observation. Formally (Reiter and De Kleer, 1987), the reasoner is given a knowledge base \( KB \) (the background theory), a set of propositional letters \( A \) (the assumption set), and a query letter \( q \). An explanation of \( q \) is a minimal subset \( E \subseteq A \) such that

1. \( KB \land (\land_{x \in E} x) \models q \) and
2. \( KB \land (\land_{x \in E} x) \neq \emptyset \).

---

\(^{9}\)The task of abduction is normally defined with arbitrary literals for explanations. For Horn theories explanations turn out to be composed of positive literals (this can be concluded from Corollary 4 in (Reiter and De Kleer, 1987)). Here we restrict ourselves to explanations composed of positive literals (by allowing only positive literals in the assumption set) when using general theories. One may therefore talk about “positive explanations” instead of explanations. We nevertheless continue with the term explanation.
Thus, abduction involves tests for entailment and consistency, but also a search for an explanation that passes both tests. We now show how one can use the algorithm from (Kautz, Kearns, and Selman, 1993) for any propositional theory $KB$.

**Theorem 4.6.1** Let $KB$ be a background propositional theory with a basis $B$, let $A$ be an assumption set and $q$ be a query. Let $B_H = \{x \in \{0, 1\}^n | \text{weight}(x) \geq n - 1\}$. Then, using the set of characteristic models $\Gamma = \Gamma_{KB}^{B_H}$ one can find an abductive explanation of $q$ in time polynomial in $|\Gamma|$ and $|A|$.

**Proof:** We use the algorithm Explain suggested in (Kautz, Kearns, and Selman, 1993) for the case of a Horn knowledge base. For a Horn theory $KB$ the algorithm uses the set $char_H(KB) = \Gamma_{KB}^{B_H}$ defined in Section 4.3. We show that adding the Horn basis $B_H$ and the additional characteristic models to a general model-based representation is sufficient for it to work in the general case.

The abduction algorithm Explain starts by enumerating all the characteristic models. When it finds a model in which the query holds, (i.e., $q = 1$) it sets $E$ to be the conjunction of all the variables in $A$ that are set to 1 in that model. (This is the strongest set of assumptions that are valid in this model.)

The algorithm then performs the entailment test ($(1)$ in the definition above) to check whether $E$ is a valid explanation. This test is equivalent to testing the deduction $KB \models (q \lor (\forall x \in E \bar{x}))$, that is a deductive inference with a Horn clause as the query. According to Theorem 4.2.2 this can be done efficiently with $\Gamma_{KB}^{B_H}$.

If the test succeeds, the assumption set is minimized in a greedy fashion by eliminating variables from $E$ and using the entailment test again. It is clear that if the algorithm outputs a minimal assumption set $E$ (in the sense that no subset of $E$ is a valid explanation, not necessarily of smallest cardinality) then it is correct. Minimality is guaranteed by the greedy algorithm, the requirement (1) by the deductive test, and the requirement (2) by the existence of the original model that produced the explanation.

It remains to show that if an explanation exists, the algorithm will find one. To prove this, it is sufficient to show that in such a case there exists a model $x \in \Gamma$ in which both the bit $q$ and a superset of $E$ are set to 1.

The existence of $x$ is a direct consequence of including the base assignment $b = 1^n$ in the basis. This is true as relative to $b$ we have $1 < b$ 0 for each bit. Therefore if there exists a model $y$ which satisfies some explanation $E$, either it is a minimal assignment relative to $b$, or $\exists x \leq y$ and $x$ is in $\Gamma$. In the first case $x = y$ is the required assignment, in the second case we observe that $y_i = 1$ implies $x_i = 1$ which is what we need.

It is quite easy to see that the above theorem can be generalized in several ways. First, we can allow the assumptions set $A$ to have up to $k$ negative literals for some constant $k$ and use the basis for $k$-quasi-Horn instead of $B_H$. Second, we can allow the query $q$ to have more than just one literal. In particular it is easy to verify that the same proof works if $q$ is a conjunction of positive literals, or if $q$ is any Horn disjunction.
4.7 Robustness of Model-Based Representations

In this section we consider a few “robustness” issues of model-based representations. In particular, we show that the compact model-based representation behaves in many respects like the complete set of models of a theory, and use it to show that it supports, in a natural way, reasoning within a changing context; this is used also to show that a model-based representation can be augmented by a (restricted) set of rules and still yield correct and efficient reasoning. We then argue that this behavior supports the incremental view of reasoning we develop in this thesis.

Reasoning within Context

It has been argued that in real life situations, one normally completes a lot of missing “context” information when answering queries (Levesque, 1986). For example if asked, while at a conference, how long it takes to drive to the airport, I would probably assume (unless specified otherwise) that the question refers to the city we are at now, Seattle, rather than where I live (and have been to the airport more times). This corresponds to assigning the value “true” to the attribute “here” for the purpose of answering the question. Sometimes we need a more expressive language to describe our assumptions regarding the current context and assume, say, that some rule applies (Selman and Kautz, 1990). For example, we may assume (in the “conference” context) that if someone has a car, then it is a rental car.

A “first principle” way to phrase this is to say that we want to deduce α from KB if α can be inferred from KB given that the query applies to the current “context”. Namely, the instances in the KB which are relevant to the query must also satisfy the (context) condition d, a conjunction of some literals and rules. We denote this question by KB |=_d α.

Notice that it is possible that KB |=_d α but KB ⊨ α, if all the satisfying assignments of KB that do not satisfy d do not satisfy α. Formalized this way, we get that the problem KB |=_d α is equivalent to the problem KB ∧ d |= α. Thus, a theorem proving approach to reasoning does not give any computational advantage in solving this reasoning problem.

Let KB ∈ F, α ∈ G and let B be a basis for G. From Theorem 4.2.5 it is clear that given Γ^BKB∧d^B, the set of characteristic models for KB∧d, model-based reasoning can be used to solve the reasoning problem KB ∧ d |= α. However, we assume here that we are given Γ^BKB and are interested in performing inference according to |=_d with it, where the “context condition” d may vary.

From our model-theoretic definition of the connective |=_d it is clear that if one holds all the models of KB, then by filtering out all the models that do not satisfy d and then performing the model-based test one can answer KB ∧ d |= α correctly. Next we show that under certain conditions this property is maintained by the compact model based representation for KB.

Given Γ^BKB, consider the following strategy to answer the reasoning problem KB |=_d α:
Algorithm \(d\)-MBR:

**Test Set:** Consider only those elements of \(\Gamma^B_{KB}\) which satisfy \(d\).

**Test:** If there is such an element which does not satisfy \(\alpha\), deduce that \(KB \not\models_d \alpha\); Otherwise, \(KB \models_d \alpha\).

Figure 4.2: \(d\)-MBR: Model-Based Reasoning within Context

The following theorem gives a simple condition for when this procedure provides correct reasoning:

**Theorem 4.7.1** Given \(\Gamma^B_{KB}\), the above procedure provides an exact solution to the reasoning problem \(KB \models_d \alpha\) for every \(d\) such that \(B\) is a basis for \(d \to \alpha\).

**Proof:** Clearly,

\[
KB \models_d \alpha \equiv KB \land d \models \alpha \\
\equiv KB \models \overline{d} \lor \alpha \\
\equiv KB \models (d \to \alpha).
\]

Therefore, from Theorem 4.2.5, when \(B\) is a basis for \(d \to \alpha\), \(\Gamma^B_{KB}\) can be used for model-based reasoning with it. Models of \(KB\) that do not satisfy \(d\) are useless as counterexamples since \(d \to \alpha\) always holds and therefore, using the Test Set of Algorithm \(d\)-MBR produces the correct inference.

Next, we use the results of Section 4.5 to give some applications of this theorem.

**Corollary 4.7.2** Let \(B\) be a basis for \(\alpha\). The following conditions on \(B\) and \(d\) satisfy the requirements of Theorem 4.7.1.

(i) Let \(\alpha\) be a Horn query and \(B = B_H\), a basis for Horn theories. If \(d\) is a monotone Boolean function then \(B\) is a basis for \(d \to \alpha\).

(ii) Let \(\alpha\) be a \(k\)-quasi-Horn query and \(B = B_{H_{2k}}\), a basis for \(2k\)-quasi-Horn theories. If \(d\) is a Boolean function that can be represented as a \(k\)-quasi-DNF (that is, a DNF representation in which there are at most \(k\) negative literals in each term) then \(B\) is a basis for \(d \to \alpha\).

(iii) Let \(\alpha\) be a \(\log n\)CNF query and \(B\) a basis for \(2\log n\)CNF theories. If \(d\) is a conjunction of up to \(\log n\) arbitrary rules (or disjunctions) then \(B\) is a basis for \(d \to \alpha\).

**Proof:** For (i), consider first the case in which \(d = \bigwedge_{i \in I} x_i\). Let \(\alpha = \bigwedge_{j \in J} (m_j \to x_j)\) be a Horn query (that is, \(m_j\) is a monotone conjunction). Then

\[
d \to \alpha \equiv \overline{d} \lor \alpha \equiv \overline{d} \lor \bigwedge_{j \in J} (\overline{m_j} \lor x_j) \equiv \bigwedge_{j \in J} (\overline{d} \lor \overline{m_j} \lor x_j) \equiv \bigwedge_{j \in J} ((d \land m_j) \to x_j),
\]

82
which means that $d \rightarrow \alpha$ is a Horn expression. In general, if $d'$ is a monotone function, $d' = \bigvee_{d_i} d_i$ where $d_i$ are monotone conjunctions. Since $\forall f, g, h, f \lor g \rightarrow h \equiv (f \rightarrow h) \land (g \rightarrow h)$, we get that $d' \rightarrow \alpha \equiv \bigwedge \lambda_{j \in J}((d_i \land m_j) \rightarrow x_j)$ is a Horn theory\textsuperscript{10}. For (ii), in the same way, we get that the body of the rule might contain up to $2k$ non-negated literals. For (iii), similar manipulations show that $d \rightarrow \alpha$ is a $2 \log n$ CNF.

The results presented in Theorem 4.7.1 and Corollary 4.7.2 can be viewed also as a process of augmenting a model-based representation $\Gamma$ with a set of rules. The results show that to allow for that, we need to maintain a model-based representation with respect to a basis that is a bit larger than the minimal basis which is sufficient to answer queries.

We have shown in this chapter the advantages of having a model-based representation, but have already hinted that the intention is to embed it as part of an inductive process that performs reasoning. The ability to allow for “rote” learning of rules in this process, and still support efficient and correct inference seems to be important and to add plausibility to this approach.

### 4.8 Concluding Remarks

This chapter develops a formal theory of model-based reasoning. We show that a simple model-based approach can support exact deduction and abduction even when an exponentially small portion of the model space is tested. Our approach builds on (1) the characterization of a set of characteristic models of the knowledge base that together capture all the information needed to reason with (2) a restricted set of queries. We prove that for a fairly large class of propositional theories, including theories that do not allow efficient formula-based reasoning, the model-based representation is compact and provides efficient reasoning. Model-based reasoning reduces to the evaluation of a propositional formula on a polynomial number of assignments, a very simple and easily parallelizable procedure.

The restricted set of queries, which we call relevant queries and common queries, can come from a wide class of propositional languages, (and include, for example, quasi-Horn theories and $\log n$ CNF), or from the same propositional language that represents the “world”. We argue that this is a reasonable approach to take in the effort to give a computational theory that accounts for both the speed and flexibility of commonsense reasoning.

The usefulness of the approach developed here is exemplified by the fact that it explains, generalizes and unifies many previous investigations and in particular, the fundamental works in automated reasoning and database theory on reasoning with Horn models (Kautz, Kearns, and Selman, 1993; Mannila and Raiha, 1986) and the

\textsuperscript{10}We note that the size of the resulting Horn theory might be exponentially large, but it only appears in the analysis. We do not actually compute this expression in the algorithm. Rather, filtering examples according to $d$ is sufficient. The same holds for the other cases.

However, we view the main interest in the model-based theory of reasoning as two fold: First, it shows that one can produce “logical” behavior even when the knowledge representation is not a logical formula and the inference procedure has nothing to do with logic or “theorem proving”. Moreover, by side-stepping the computational difficulties introduced by the logical representation, we can perform efficient reasoning in cases in which formula-based reasoning is intractable. We believe that this could be an important lesson when we try to model the phenomena we discuss here on other, possibly biologically plausible, computational models.

Second, model-based representations support an incremental view of reasoning in a natural way. We saw in Section 4.7 that, given a model-based representation, the model-based approach to the general reasoning problem can be used to reason within a narrower context defined by a context condition $d$. We call this a top-down solution. It is conceivable, though, that an agent would have only some of the models, those models of $KB$ that satisfy some specific context condition $d$. In such a case, our results show that the agent reasons correctly within this context (although she cannot reason within every context).

Similarly, as in Theorem 3.2.1, if an agent has access to some oracle which supplies examples from a context distribution $D$, then it can reason correctly within this context. As discussed in the previous chapter, the availability of these oracles, which enable tractable reasoning, seems plausible assuming that the intelligent agent interacts with the environment, as we do in this thesis. This discussion supports the view that an intelligent agent constructs her view of the world incrementally by pasting together many “narrower” views from different contexts.

We have shown in this chapter the advantages of having a model-based representation, and have already hinted that the intention is to embed it as part of an inductive process that performs reasoning. We believe that the ability to allow for “rote” learning of rules in this process, as we have shown to be the case in Section 4.7, while still supporting efficient and correct inference, is important and adds plausibility to this approach.

In the next chapter we make the inductive view of reasoning more explicit when we further develop the Learning to Reason framework and illustrate the importance of the model-based approach for this framework.
I do not wish, at this stage, to examine the logical justifications of this form of argumentation; for the present, I am considering it as a practice, which we can observe in the habits of men and animals.

Bertrand Russell, Philosophy, 1927

In this chapter we gather the fruits planted in the previous two chapters and exhibit the computational advantages of the Learning to Reason approach. We deviate from the traditional setting of “first learn to classify, then reason with the hypothesis”: a learning algorithm is used first, but rather than learning a “classifying hypothesis”, it constructs a knowledge representation that allows for efficient reasoning.

We prove the usefulness of the Learning to Reason approach by showing that through interaction with the world, the agent truly gains additional reasoning power over what is possible in the traditional setting.

First, we develop Learning to Reason algorithms for classes of propositional languages for which there are no efficient reasoning algorithms, when represented as a traditional (formula-based) knowledge base. In line with the Learning to Reason framework, these algorithms, that aim at different tasks (i.e., classes of queries), use different intermediate knowledge representations. Second, we develop Learning to Reason algorithms for a class of propositional languages that is not known to be learnable in the traditional sense.
5.1 Learning to Reason without Reasoning

We present Learning to Reason algorithms for classes of propositional languages for which there are no efficient reasoning algorithms, when represented as a traditional (formula-based) knowledge base. We present two orthogonal results which trade the expressivity of the world with the expressivity of the class of queries we can answer. These results provide a nice illustration of the Learning to Reason framework in that each algorithm makes use of a different intermediate knowledge representation. First we present the result that makes use of a model-based representation to side-step the difficulty of reasoning with the traditional, formula-based representation. The result presented in the next subsection uses the notion of a least-upper-bound of a theory with respect to Horn functions to achieve this goal.

5.1.1 Model-Based Representation

Theorem 5.1.1 There is an Exact-Learn to Reason algorithm, that uses an Equivalence Query oracle and a Membership Query Oracle for the reasoning problem \((CNF \cap DNF, Q)\), where \(Q\) is any class of relevant and common queries.

Proof: The Learning to Reason algorithm learns a model-based representation for the target function \(f\) and then uses model-based reasoning to answer queries with respect to it. In (Bshouty, 1993a) an algorithm is developed that uses an Equivalence Query oracle and a Membership Query Oracle to learn an exact representation of any function \(f \in CNF \cap DNF\). As a byproduct of this algorithm, the set of all minimal models with respect to a basis \(B\) of \(f\) is produced. Using this set of models \(\Gamma\), model-based reasoning, via Theorem 4.2.1, gives the result for relevant queries. Also, given basis \(B\), the algorithm developed in (Bshouty, 1993a) produces as a byproduct the set of minimal models of \(f\) with respect to \(B\), provided that \(f\) has small DNF. Combining this with Theorem 4.2.5 we get the result for common queries.

Notice that since the above algorithm learns a model-based representation for the target function, using the results of Section 4.6 we can get in exactly the same way an algorithm that performs abduction.

The above theorem is an example for a reasoning problem that is provably hard in the “traditional” sense and has an efficient solution in the new model. Given a CNF knowledge base, even with the added information that it has a short DNF, the reasoning problem is still hard. This is so since it is NP-hard to find a satisfying assignment for a CNF expression even if one knows that it has exactly one satisfying assignment (Valiant and Vazirani, 1986). In this case, the additional reasoning power of the agent is gained through the interaction with the world by using \(EQ(f)\).
5.1.2 Horn Queries

In the previous section we have shown that for various worlds we can learn to reason with respect to relevant or common queries. We give here an orthogonal result that shows that we can learn to reason from every Boolean function (with polynomial size Horn-least upper bound), with respect to Horn queries. The result is based on an algorithm developed by Frazier and Pitt (1993) for the purpose of learning to classify Horn theories.

Another difference from the previous section is that the oracles used here, following Frazier and Pitt (1993), are entailment oracles. The agent “learns” the world from examples that are statements the world entails and does not entail, rather than, say, seeing positive and negative instances of the world, as in $EX_D(f)$ and $MQ(f)$. The oracles are formally defined in Section 3.1.2.

**Theorem 5.1.2** There is an Exact-Learn to Reason algorithm, that uses an Entailment Equivalence Query oracle and an Entailment Membership Query Oracle, for $(\mathcal{F}, H)$, where $\mathcal{F}$ is the class of all Boolean functions with polynomial size Horn least upper bound.

**Proof:** Although it is not presented in exactly this form, the results of Frazier and Pitt (1993) (algorithm LRN) immediately give an algorithm that finds the Horn least upper bound of any Boolean function $f$ using entailment queries. As was shown in Theorems 4.2.4 and 4.2.5, when reasoning with respect to Horn queries, it is sufficient to hold a Horn-approximation of the theory and therefore reasoning with the outcome of the algorithm LRN provides exact deduction. Since that output of the algorithm LRN, namely, the LUB of the theory with respect to Horn, is represented as a Horn function, this procedure is polynomial in the LUB size.

In particular, this provides another example in which there exists an efficient L2R algorithm but reasoning is hard, similar to Theorem 5.1.1. In this case, the problem $f \models \alpha$, even when $\alpha$ is a Horn query, is NP-hard if $f$ is given as a CNF formula.

5.2 Learning to Reason without Learning to Classify

In this section we present two results on Learning to Reason any Boolean function $f$ with a polynomial size DNF. These results are significant since there is no known algorithm that Learns to Classify DNF. We present two algorithms. The first algorithm (Theorem 5.2.1) makes use of a Reasoning Query Oracle $RQ(f, Q)$ and a Membership Query Oracle $MQ(f)$ to exactly Learn to Reason for the problem $(f, Q)$. It is worth noticing that this algorithm exhibits an Exact-Learning to Reason algorithm even though its knowledge base consists of an approximate description of the world. The second, more complicated algorithm, uses a “weaker”
interface, $EX_D(f)$ instead of $RQ(f, Q)$, and yields a PAC-Learn to Reason algorithm for $f$. The main idea in both algorithms is that it is sufficient to learn the least upper bound of a function in order to reason with common queries.

It is interesting to note that in some sense, a PAC-Learn to Reason algorithm for any Boolean function is implied from the one-time sampling approach developed in Chapter 3. However, the version presented here has two additional properties. First, the sampling complexity of the algorithm in Theorem 5.2.3 depends on the size of the class $\mathcal{F}$ which represents the world, as opposed to the one-time sampling, where it depends on the size of the class $\mathcal{Q}$, from which the queries are taken. Second, and more significant, is the fact that the algorithm in Theorem 5.2.3, while guaranteeing only pac behavior, has the property that with a bounded number of reasoning mistakes it converges to yield exact reasoning performance. We believe that this property is useful for a Learning to Reason algorithm.

We now present the algorithms. Both are based on a modified version of Bshouty’s algorithm to learn Boolean functions via the monotone theory (Bshouty, 1993a).

### 5.2.1 Exact Learning to Reason

Let $B$ be a basis for the class of queries $\mathcal{Q}$. The algorithm collects a set of models $\Gamma = \bigcup_{b \in B} \Gamma_b$, the set of locally minimal assignments of $f$ with respect to $B$. Since this set contains the set of minimal models of $f_{ub}$ with respect to the basis $B$ it guarantees, by Theorem 4.2.5, exact reasoning with respect to queries in $\mathcal{Q}$. The algorithm $EX-L2R-DNF$ is depicted in Figure 5.1. The following theorem proves its correctness.

**Algorithm Ex-L2R-DNF**

1. $\forall b \in B$, initialize $\Gamma_b \leftarrow \emptyset$.
2. $\alpha \leftarrow RQ(f, Q)$
3. Answer $f \models \alpha$ by performing model-based test on $\Gamma = \bigcup_{b \in B} \Gamma_b$.
4. If “wrong” then
5. let $x$ be the counterexample received from $RQ(f, Q)$
6. $\forall b \in B$ such that $x \not\in M_b(\Gamma_b)$
7. $\Gamma_b \leftarrow \Gamma_b \cup \text{Find-min-model}(x, b)$
8. GoTo 2

Figure 5.1: The Algorithm $EX-L2R-DNF$
Procedure Find-min-model($x, b$)

1. If $\exists y \in [x]_b$ such that $f(y) = 1$ /* use $MQ(f)$ */
2. $x \leftarrow y$ ; GoTo 1
3. Else
4. Return($x$)

Figure 5.2: The Procedure Find-min-model($x, b$)

**Theorem 5.2.1** Algorithm $EX$-$L2R$-$DNF$ is a MB-Learn to Reason algorithm for the problem $(DNF, \mathcal{Q})$, where $\mathcal{Q}$ is the class of all common queries.

**Proof:** Denote $h = \land_{b \in B} (\lor_{z \in \Gamma_b} \mathcal{M}_b(z))$. Clearly, at every step in the algorithm, $\Gamma \subseteq f$, and therefore the algorithm $EX$-$L2R$-$DNF$ never makes a mistake when it says “no” (and is therefore well defined). Whenever the algorithm errs on an $RQ(f, \mathcal{Q})$ query $\alpha$, it receives from the oracle a positive counterexample, $x \in f \setminus \alpha$. We first argue that $x \in f \setminus h$. In order to show that, we prove that $h \subseteq \alpha$. Indeed, let $y$ be such that $h(y) = 1$. Then, $\forall b \in B, \exists z_b \in \Gamma_b$ such that $z_b \subseteq y$. Since the response on the reasoning query was “yes”, we have that in particular $\alpha(z_b) = 1$, for all those points $z_b$. Now, since $\alpha \in \mathcal{Q}$, using Corollary 4.1.4 we get that $\alpha(y) = 1$.

Now, since $x \notin h$, it is negative for at least one of the $b$’s in the conjunction defining $h$. Therefore, there exists a model $z \in \min_{\mathcal{B}}(f) \setminus \Gamma_b$ for each such $b$. Therefore, in this case, the algorithm can use the procedure $Find-min-model(x, b)$ to find a new model of $f$, an element of $\min^*_b(f)$. $Find-min-model(x, b)$ is a standard procedure (Angluin, 1988; Bshouty, 1993a) that uses a sequence of calls to $MQ(f)$ to find an element of $\min^*_b(f)$. Since for all $b \in B$, $|\min^*_b(f)| \leq |DNF(f)|$, the size of this representation is polynomial. Moreover, $\Gamma^B_j \subseteq \cup_{b \in B} min^*_b(f) \subseteq f$.

The algorithm might make reasoning mistakes as long as there is an element of $\Gamma^B_j$ missing from its model based representation, but with every such mistake it makes progress toward collecting the elements in the set $\Gamma^B_j$. Therefore, after at most $|\cup_{b \in B} min^*_b(f)| \leq |B| \cdot |DNF(f)|$ mistakes algorithm $EX$-$L2R$-$DNF$ makes no more mistakes on $RQ(f, \mathcal{Q})$ queries and by Theorem 4.2.4, $h = f_{\text{ub}}$.

Therefore, Theorem 4.2.5 implies that $EX$-$L2R$-$DNF$ guarantees exact reasoning on queries from $\mathcal{Q}$.

### 5.2.2 PAC Learning to Reason

The algorithm $PAC$-$L2R$-$DNF$ collects a set of locally minimal models of $f$ with respect to a fixed basis $B$, a basis for $\mathcal{Q}$. By Theorem 4.2.4 this yields a representation of the least upper bound of $f$ in $\mathcal{Q}$, and by Theorem 4.2.5, this is enough to solve the $(\mathcal{F}, \mathcal{Q})$ reasoning problem. Unlike the exact $L2R$ algorithm, $EX$-$L2R$-$DNF$, here we simulate the $RQ(f, \mathcal{Q})$ oracle by calling the Example Oracle.
Algorithm PAC-L2R-DNF

Learning Phase:

1. $\forall b \in B$, initialize $\Gamma_b \leftarrow \emptyset$.
2. Let $h = \land_{b \in B} (\lor_{z \in \Gamma_b} \mathcal{M}_b(z))$.
3. Call $EX_D(f)$ $m = \frac{1}{\epsilon} \log \frac{1}{\delta}$ times. Let $S$ be the set of all $m$ samples.
4. If $\exists x \in S$ such that $f(x) = 1$ and $h(x) = 0$ GoTo 9 /* no positive counterexample */
5. $\forall x \in S$ such that $f(x) = 1$ and $h(x) = 0$ do: /* positive counterexample */
6. $\forall b \in B$ such that $x \notin \mathcal{M}_b(\Gamma_b)$ do:
7. $\Gamma_b \leftarrow \Gamma_b \cup \text{Find-min-model}(x, b)$
8. GoTo 2
9. Return $\Gamma = \bigcup_{b \in B} \Gamma_b$

Reasoning Phase:

Answer queries by performing model-based reasoning on $\Gamma = \bigcup_{b \in B} \Gamma_b$

Figure 5.3: The Algorithm PAC-L2R-DNF

$EX_D(f)$ sufficiently many times. This simulation builds on the standard simulation of Equivalence Query Oracle by calls to Example Oracle (Angluin, 1988) and uses some nice properties of the reasoning with models framework. On the other hand, this algorithm is not a mistake-bound algorithm, and the time it takes to achieve its final state does not depend on the behavior of the oracles and can be bounded.

Figure 5.3 describes the algorithm PAC-L2R-DNF. For all $b \in B$ we initialize $\Gamma_b = \emptyset$ and maintain a hypothesis $h = \land_{b \in B} (\lor_{z \in \Gamma_b} \mathcal{M}_b(z))$. To get counterexamples, we simulate $EQ(f)$ by $m = \frac{1}{\epsilon} \log \frac{1}{\delta}$ calls to $EX_D(f)$. On a positive counterexample the algorithm looks for a locally minimal assignment of $f$ to be added to $\Gamma_b$. Such an assignment always exists since this example is negative for at least one of the $b$’s in the intersection of $h$. Finding the locally minimal assignment is done by calling the procedure $\text{Find-min-model}(x, b)$ that uses $MQ(f)$ greedily to find a locally minimal element. That is, the algorithm increases the size of at least one of the $\Gamma_b$’s. Whenever the algorithm receives a negative counterexample, it simply ignores it. The algorithm stops if $m$ consecutive calls to $EX_D(f)$ do not return any positive counterexample.

We note that simulating $EQ(f)$ by $EX_D(f)$ is essential to our algorithm. Oth-
otherwise, an adversarial equivalence oracle can adapt its strategy to the hypothesis that the algorithm holds. In particular, it could prevent the algorithm from making progress by presenting the same negative counterexample forever. The following lemma shows that with an example oracle this is not the case.

**Lemma 5.2.2** Let \( f \) be a Boolean function with a polynomial size DNF representation and let \( f_{\text{ub}} \) be its least upper bound in \( Q \). Then algorithm PAC-L2R-DNF runs in time polynomial in \( n \) and, with probability \( > 1 - \delta \) reaches the reasoning phase with a hypothesis \( h \) which has the following properties: (1) \( h \subseteq f_{\text{ub}} \) (2) \( \text{Prob}_D[f \setminus h] < \epsilon \).

**Proof:** Property (1) easily follows from Theorem 4.2.4 and from the fact that at any point the algorithm holds in \( \Gamma_b \) only (locally) minimal assignment of \( f \).

For (2), we first note that the size of \( \Gamma_b \) for any basis assignment is bounded by the DNF size of \( f \), and is therefore polynomial. This implies that there is a polynomial bound on the number of positive counterexamples that the algorithm might receive. Now consider the algorithm’s hypothesis when it stopped. If the algorithm already used up its mistake bound then by Theorem 4.2.4, \( h = f_{\text{ub}} \) and we are done. Otherwise, it stopped because it did not receive a positive counterexample from the simulation of the equivalence query. By the selection of \( m \), this implies that with probability \( 1 - \delta \), \( \text{Prob}_D[f \setminus h] < \epsilon \).

The following theorem shows that the properties of Lemma 5.2.2 guarantee the correctness of the algorithm.

**Theorem 5.2.3** Algorithm PAC-L2R-DNF is a PAC-Learn to Reason algorithm for the problem \((\text{DNF}, Q)\), where \( Q \) is the class of all common queries.

**Proof:** Let \( \Gamma = \bigcup_{b \in B} \Gamma_b \) be the output of the learning phase of the algorithm. \( \Gamma \) is a subset of the set of all locally minimal assignments of \( f \) with respect to \( B \).

To prove the correctness of the algorithm consider first the case in which \( f \models \alpha \). Since \( \Gamma \subseteq f \), \( \forall x \in \Gamma, \alpha(x) = 1 \) and the algorithm answers “yes”.

Assume now that \( f \not\models \alpha \), and suppose, by way of contradiction, that algorithm answers “yes” on \( \alpha \). Since \( \alpha \) is \((f, \epsilon)\)-fair, we know that \( \text{Prob}_D[f \setminus \alpha] > \epsilon \). Since by Lemma 5.2.2, \( \text{Prob}_D[f \setminus h] < \epsilon \), this implies that \( h \setminus \alpha \neq \emptyset \).

Let \( y \in h \setminus \alpha \). Since, by the assumption, the algorithm answers “yes”, we know that for all \( u \in \Gamma \), \( \alpha(u) = 1 \). Since \( B \) is a basis for \( \alpha \) and for all \( u \in \Gamma \), \( \alpha(u) = 1 \), Corollary 4.1.4 implies that

\[
\forall u \in \Gamma, \forall b \in B, \exists v_{u,b} \in \min_b(\alpha), \text{ such that } u \geq_b v_{u,b}.
\] (5.1)

Using the fact \( y \in h = \land_{b \in B} (\lor_{z \in \Gamma_b} M_b(z)) \), we have that

\[
\forall b \in B, \exists z \in \Gamma_b, \text{ such that } y \geq_b z.
\] (5.2)
By the assumption, all the elements $z$ identified in Equation 5.2 satisfy $\alpha$ and therefore, as in Equation 5.1 we have that

$$\forall z \in \Gamma, \forall b \in B, \exists v_{z,b} \in \min_b(\alpha) \text{ such that } z \geq_b v_{z,b}. \quad (5.3)$$

Substituting Equation 5.3 into Equation 5.2 gives

$$\forall b \in B, \exists v_{(z),b} \in \min_b(\alpha) \text{ such that } y \geq_b v_{(z),b}$$

which implies, by Corollary 4.1.4, that $\alpha(y) = 1$, a contradiction.

We have shown that there exist Learning to Reason algorithms for all the Boolean functions with a polynomial size DNF, provided that the queries are common. This should be contrasted with the inability to learn to classify DNF. One can learn $f_{\text{ab}}$ and reason with it with respect to common queries, but $f_{\text{ab}}$ is not sufficient as a substitute for $f$ when classifying examples, since we cannot bound the size of $\text{Prob}_D[h \setminus f]$.

### 5.3 Concluding Remarks

In this chapter we have shown that the approach developed so far in this thesis indeed provides a true increase in the reasoning power of the agent.

Our definition of the new framework was aimed at overcoming the main computational difficulties in the traditional treatment of reasoning. Indeed, we have shown that when learning is an integral part of the process, and when we do not constraint the intermediate knowledge representation but rather find the one that suits the reasoning task most, we can provide reasoning power that earlier theories can not. This is even more significant when we exhibit cases in which the new framework allows for successful algorithms, but stated separately, neither the reasoning problem nor the learning problem are efficiently solvable.

So far, we have discussed only deductive and abductive reasoning, and clearly, other reasoning tasks should also be addressed. Moreover, while this framework stresses the importance of the type of interface between the agent and her environment, so far we have made fairly strong assumptions on the interface. In particular, we have assumed that when given an example, all the attributes of the world are assigned values in it.

In the next chapter we investigate the interface issue further and study, within the Learning to Reason framework, the case in which the interface provides the agent only partial information.
Earlier formalisms in the knowledge-based systems framework have abstracted the reasoning task as a deduction task, of determining whether a sentence, assumed to capture the situation at hand, is implied from the knowledge base, capturing our “theory” of the world. This abstraction has been criticized by many (e.g., (Minsky, 1975)) on the ground that it cannot support non-monotonic reasoning.

Indeed, it is widely acknowledged today that a large part of our everyday reasoning involves arriving at conclusions that are not logically entailed by our “theory” of the world. Many conclusions are derived in the absence of sufficient information to deduce them. This type of reasoning is naturally non-monotonic, since further evidence may force us to retract the conclusions.

In light of this, many researchers have tried to augment the knowledge base and to modify the inference mechanisms so as to allow reasoning in the presence of incomplete information. The idea is to augment the true knowledge (facts and rules) we have about the world with a set of assumptions that capture only “typical” cases. These assumptions are called default assumptions, or simply defaults. Within the knowledge-based systems approach, defaults are stored in the knowledge base along with the other, non-default, knowledge. The quest is for a reasoning system that, given a query, responds in a way that agrees with what we know about the world and some subset of the default assumptions and at the same time supports our intuition about a plausible conclusion. The process of reasoning with the knowledge and the defaults, is called default reasoning, and numerous formalisms that attempt at acceptable reasoning behavior have been studied for it. (e.g., (AI, 1980; Touretzky, 1986; McCarthy, 1986; Reiter, 1987; Etherington, 1988; Goldszmidt and Pearl, 1991; Pearl, 1988; Geffner, 1990)).

As argued previously in the thesis, computational considerations apparently render all the formalisms suggested within the knowledge-based-system approach
inadequate for common-sense reasoning. The hardness results are especially significant in the case of non-monotonic reasoning, where the various formalisms have been shown to be even harder to compute than the corresponding deduction tasks (e.g., (Kolaitis and Papadimitriou, 1988; Kautz and Selman, 1991; Selman, 1990; Papadimitriou, 1991)). The increase in complexity is clearly at odds with the intuition that reasoning with defaults somehow reduces the complexity of reasoning.

Moreover, some studies of reasoning in the presence of incomplete information within the knowledge-based systems framework have shown that capturing even what people view as plausible patterns of reasoning is not easy (see, e.g., (Touretzky, Hory, and Thomason, 1987)). Most formalisms, in attempting to capture some aspects of “default” reasoning give up on others. Multiple levels of specificity of information, irrelevant information and conflicting defaults are among the aspects that the various formalisms have found difficult to reconcile.

**Handling Incomplete Information**

As we argue throughout this thesis, commonsense reasoning, and in particular, reasoning in the presence of incomplete information, is an inductive phenomenon; when the notion of consistency is at the heart of the formal reasoning system, as in most previous approaches, inductive phenomena are difficult to capture.

In previous chapters we have introduced a new framework for the study of reasoning, which incorporates a role for inductive learning within efficient reasoning. We have argued there (Section 3.5.2), on qualitative grounds, that reasoning with an inductively learned knowledge base, as in the Learning to Reason framework, exhibits a desirable non-monotonic reasoning behavior as a side effect, by using reasoning mistakes to improve the knowledge base.

In this chapter we extend the Learning to Reason framework to deal explicitly with reasoning in the presence of incomplete information. In line with the overall approach presented here, we present the view that the world is very complicated and there is no hope of acquiring an exact representation of it; our aim should be to acquire enough information with which to cope effectively in the world. In doing so we extract certain regularities from the world, and assume that in similar circumstances we can rely on these.

Consider, for example, concluding from the knowledge that Tweety is a bird that Tweety can *any*. This conclusion is useful, and is clearly justified in some situations, e.g., when discussing birds in Boston during their migration season. A different conclusion to the same query will be suggested, though, by a veterinarian working in a birds’ hospital, or by someone raised in an ostrich nature reserve. Of course, the possible circumstances in which any “presumed” correct line of reasoning can be defeated astound, and we are doomed to make mistakes when our experience does not support the current situation.

The key to the approach we develop is the view that regularities occur not only in what we observe (e.g., if all elephants we have seen had a trunk, we might think that all elephants have a trunk) but also in what we do not observe (e.g., if in previous
experience of flying birds we were not aware to whether they were penguins, maybe unawareness to penguin-ness of birds indicates that they fly.

That is, missing information in the interaction of the agent and her environment may be as informative as observed information. In this chapter we formalize this intuition and use it to develop a theory that supports efficient reasoning with incomplete information.

Our treatment of incomplete information follows the suggestion made in (Valiant, 1994b). While there, in an effort to formalize the notion of Rationality, Valiant presents a more comprehensive view of the phenomena that comprise cognition, here we present a more detailed account of reasoning in the presence of incomplete information focusing on presenting it as a problem of Learning to Reason.

In our framework the intelligent agent is given access to her favorite learning interface; by interaction with the environment she inductively learns her representation of the world that is used later to respond to queries. Her performance on the reasoning task is measured in a way that makes explicit the dependence of the reasoning performance on the input from the world. We assume an “on-line” model of learning, in which reasoning mistakes are used to improve the representation and thus affect future reasoning behavior.

Unlike previous theories of reasoning in the presence of incomplete information, we are not interested in providing a theory of defaults, but rather a theory of inference. Defaults do not exist in the world. One cannot evaluate whether a default is true in the world but only if a conclusion derived by a reasoning system is acceptable. The representation developed here provides a richer language for dealing with reasoning problems. In particular, we show that many default reasoning scenarios with which previous formalisms have struggled have concise representations in our framework; moreover, since these representations can be learned efficiently from interaction with the environment, this yields efficient Learning to Reason algorithms.

We study in this chapter the problem of Learning to Reason from incomplete information within a knowledge representation which we call the attribute functions representation. While we show that this specific knowledge representation is not crucial to the approach, which can be studied also in the more general representations studied previously in the thesis, we prefer to discuss the attribute-function representation here since this may be a better starting point for a more formal investigation of these questions in other models and in particular, in Valiant’s neurولدal model (Valiant, 1994a).
In the rest of this chapter we go beyond the view that dealing with incomplete information is a "necessary evil", and formalize the intuition that incomplete information may actually help in supporting plausible reasoning efficiently. After presenting the framework in Section 6.2 we illustrate, in Section 6.3, how various problems in reasoning with defaults are dealt with in our framework. In Section 6.4 we discuss some of the learning issues that this framework raises and in Section 6.5 we justify the reasoning task and representation we consider and relate it to other reasoning tasks. We conclude by discussing the results and some theoretical and empirical questions our approach raises.

6.1 First Attempt

So far we have assumed, as is common in computational learning theory, that in examples observed by the agent values are assigned to all the attributes in the world. This may be reasonable in many learning to classify tasks, such as character recognition. However, when learning to reason, in many cases one cannot assume that all the attributes in the world are assigned values. Rather, the information perceived provides only partial information on the state of the world. For example, when sitting in a windowless lecture hall one’s senses do not supply any information about the current weather. Some of the missing information might be irrelevant to the task at hand but some might be relevant.

In this section we extend the framework and some of the results to the case in which the interaction of the learner with the world supplies partial information. We show how the methodology developed previously in this thesis can be extended to yield some interesting positive results, even when the interaction with the world is via partial assignments. In particular, we exhibit in this case also a Learning to Reason algorithm for a reasoning problem which is intractable in the traditional setting.

In this section we still assume, as in previous chapters, that there is a consistent world, $W : \{0, 1\}^n \rightarrow \{0, 1\}$, and we therefore need to discuss what meaning can be attributed to partial observations.

We consider a set $X = \{x_1, \ldots, x_n\}$ of Boolean variables, each of which can take the values 1 or 0 to indicate whether the associated world’s attribute is true or false. A vector (partial example, partial assignment) assigns to each of the $n$ variables a value from $\{0, 1, *\}$. The symbol $*$ denotes that a variable is unknown. For example $v = (1, *, 0)$ means that $x_1$ is true, $x_3$ is false, and the value of $x_2$ is unknown. A vector is total (total example, total assignment) if every variable is known (i.e., assigned value from $\{0, 1\}$).

Consider a positive example $v \in \{0, 1, *\}^n$ which is only partially specified.

---

1This approach is further developed in (Khardon and Roth, 1994c) where other results in this line are presented.
There are various ways in which we can interpret the meaning that \( v \) conveys on \( W \). Consider, for example, the following two interpretations:

1. Universal interpretation: For all possible extensions \( v' \) of \( v \) to total vectors, \( W(v') = 1 \).

2. Existential interpretation: There exists an extension \( v' \) of \( v \) to a total vector, such that \( W(v') = 1 \).

Approach (1) is taken in (Valiant, 1984) to model the fact that the agent receives all the attributes that are relevant for the concept being learned. This is more suitable for the task of learning to classify a single concept, where the learner is presented positive or negative examples of the target concept. For the task of learning a "world" representation in order to reason about it later, it seems that the agnostic approach, the existential interpretation (2), is preferable.

A motivating scenario is that of an agent who is wandering around in the world, but can perceive at any instance only a limited number of attributes. In general, the agent has no control of the perceived attributes, nor can she tell if all the "important" attributes have been perceived. This situation can be modeled by having someone "hide" some of the attributes in an example. As we show later even with this seemingly pessimistic interpretation some interesting positive results can be proved.

### 6.1.1 Learning to Reason without Reasoning: The Partial Assignments Case

In order to model the agent’s interface with the world we use the oracles defined in Chapter 3. We use the same notation, except that now we allow the use of partial assignments, \( x \in \{0, 1, *\}^n \). When we write \( f(x) \), where \( f \) is a Boolean function and \( x \in \{0, 1, *\}^n \) a partial assignment, we mean the value of \( f_e \) or that of \( f_u \) on \( x \), where \( f_e \) and \( f_u \) are evaluated based on the existential or universal interpretations defined above, respectively. Note that, when defining the example oracle \( EX_D(f) \), \( D \) is now a probability distribution over \( \{0, 1, *\}^n \).

The partial assignment interpretations defined above have some very interesting properties with respect to logical implication, function evaluation etc., that we do not discuss here (See (Khardon and Roth, 1994c) for details.). In this section we present only one example, in order to illustrate a case of “Learning to Reason without Reasoning”, and to contrast this approach with the approach developed in the rest of the chapter.

Assume we would like to learn to reason with functions \( f \in k\text{CNF} \). Consider first the problem of learning \( f \in k\text{CNF} \), given access to \( EX_D(W) \), where \( D \) is a distribution over \( \{0, 1, *\}^n \). It is not hard to see that a variation of the well known algorithm for \( k\text{-CNF} \) (Valiant, 1984) works in this case as well. We start by listing all \( 3^k \binom{n}{k} \) disjunctions. Let the hypothesis \( h \) be the conjunction of all these disjunctions.
Then, when given a positive (partial) example \( x \) drawn according to \( D \), on which our hypothesis is evaluated to 0, we erase the disjunctions which are evaluated to 0. It is easy to see that this yields a mistake-bound and therefore a pac-learning to classify algorithm for \( f \), for the abbreviated and universal interpretations. For the existential interpretation, where evaluating a CNF on a partial assignment is computationally hard, the above algorithm is not efficient. If, however, we evaluate the hypothesis on partial examples in a “lazy” manner, the algorithm works for the existential interpretation as well. According to the “lazy” evaluation, \( h(v) = 0 \) if and only if one of the clauses in \( h \) is falsified by \( v \). Now, using the same algorithm as above, when given a positive partial example \( x \), evaluate each of the disjunctions on \( x \), and eliminate the disjunction \( d \) from the list if and only if \( d(x) = 0 \). As before, it is easy to see that this yields a mistake-bound and therefore a pac-learning to classify algorithm for \( f \), for the existential interpretation as well.

Suppose, however, that the purpose of learning was to perform deductive reasoning with the hypothesis. In this case, given a query \( \alpha \in k\text{CNF} \), it is intractable to answer \( f \models \alpha \).

Thus, we can observe here the same mismatch between learning and reasoning as we have pointed out in Chapter 3. As we did in previous chapters, we will investigate model-based representations, in this case, partial assignments based representations, and see if this can help and lead to a Learning to Reason algorithm.

**Reasoning and Partial Assignments**

We have shown in Chapter 4 that we can answer queries using a model-based representation that consists of total assignments. It is now natural to ask whether this is possible when the representation consists of only partial assignments. For the rest of this section we concentrate on the existential interpretation of partial assignments.

Let \( \Gamma_f \) be the (polynomial size) set of characteristic models, guaranteed by Theorem 4.5.6, that supports correct model based reasoning with queries from \( k\text{CNF} \). (Notice that the theorem guarantees a set of characteristic models for every function \( f \), but this set depends on the class of queries we want to answer). Let the query \( \alpha \) be a disjunction on \( k \) literals. In the model-based approach, in order to decide if \( f \models \alpha \), we need to evaluate \( \alpha \) on each of the elements of \( \Gamma_f \). Since assignments to the \( k \) variables in \( \alpha \) are sufficient to evaluate a disjunction \( \alpha \), we need only consider, for every subset of size \( k \) of the variables, the projections of each of the elements of \( \Gamma_f \) on it. Moreover, it is easy to see that in order to guarantee correct deduction on all such queries, we need to consider all the subsets of size \( k \), and the projections of all characteristic models on those sets. Otherwise, we can find a query on which the model-based reasoning will not answer correctly. In the next paragraph we make this discussion more formal, for the case of \( k\text{CNF} \) queries.

For any \( k \), there are \( \binom{n}{k} \) subsets of size \( k \) of the \( n \) variables. Given an element \( x \in \{0,1\}^n \) and a subset \( I \) of \( k \) variables, the projection of \( x \) on \( I \) is the partial vector \( v \) defined by: \( v_i = x_i \), for all \( x_i \in I \), and \( v_i = * \) otherwise. Projecting all
the elements of $\Gamma_f$ on each of these subsets we get a set of $|\Gamma_f|\binom{n}{k}$ partial vectors. We denote this set by $\Gamma^k_f$. We now define a version of the model-based reasoning algorithm, that is useful when working with the set $\Gamma^k_f$ of partial assignments.

The lazy model-based reasoning algorithm for CNF queries works as follows. The algorithm keeps partial assignments in its knowledge base. When it receives a CNF query $\alpha$, the algorithm checks whether one of the partial assignments in its knowledge base falsifies one of the clauses in $\alpha$. If it finds such a partial vector, it says ‘no’ and otherwise it says ‘yes’.

**Algorithm LAZY-MBR**

*Test Set:* A set $\Gamma$ of partial satisfying assignments.

*Test:* Given a CNF query $\alpha$, if there is an element $x \in \Gamma$ which falsifies one of the clauses in $\alpha$, deduce that $f \not\models \alpha$; Otherwise, $f \models \alpha$.

Figure 6.1: LAZY-MBR: Model-Based Reasoning with Partial Assignments

Note that the algorithm is slightly different from the normal model-based Algorithm MBR. The latter, when given a query and an assignment, will try to evaluate the query on this assignment to test whether the assignment satisfies the query. Since this may be a hard task for partial assignments, the lazy algorithm only tests for direct falsification of clauses in the query, and otherwise gives up. As the following theorem shows, if the knowledge base contains the right partial assignments, the lazy algorithm is guaranteed to succeed.

**Theorem 6.1.1** The lazy model-based reasoning algorithm, when using the set $\Gamma^k_f$, is correct for all queries $\alpha \in kCNF$.

**Proof:** Clearly, if $f \models \alpha$, model-based reasoning answers correctly. Assume therefore that $f \not\models \alpha$. Theorem 4.5.6 implies that there exists a model $z \in \Gamma_f$ such that $\alpha(z) = 0$. In particular, since $\alpha$ is a $kCNF$, one of it clauses $C_z$ must be falsified by $z$, That is $C_z(z) = 0$. Now consider the element $z'$ which is the projection of $z$ on the variables in $C_z$. Clearly $z'$ is in $\Gamma^k_f$ and $C_z(u) = 0$.

The previous theorem shows that given $\Gamma_f$ we can generate a new model based representation, $\Gamma^k_f$, by projecting on all possible subsets of size $k$ of the variables. In (Khardon and Roth, 1994c) this problem, as well as other problems in this approach are discussed further and in particular, it is shown that this is essentially the most one can do with partial assignment-based representations.

\[^2\text{Note that, according to its construction, these partial assignments should be considered using the existential interpretation.}\]
Learning to Reason: Partial Assignments-Based Representations

As we have shown in Chapter 3, a sampling approach to reasoning can be used in this case as well. Given access to $EX_D(f)$, where $D$ is a probability distribution over $\{0, 1, *\}^n$, the same algorithm used there can answer correctly a large class of queries (with the same restrictions and caveats discussed there).

In this section we present algorithms that can support exact Learning to Reason with $k$CNF queries.

We first present an on-line algorithm that can learn to reason with $k$CNF queries. The Algorithm $A$ interacts with $RQ(F, Q)$ (see Def. 3.1.7) and simply collects all the counterexamples it receives into a set $G$. In order to answer the queries, the algorithm performs lazy model-based reasoning with the set $G$. In some sense, this algorithm can be seen as an on-line version of the sampling algorithm.

**Theorem 6.1.2** The algorithm $A$, when given access to $RQ(F, kCNF)$, is a mistake bound learn to reason algorithm for the reasoning problem $(F, kCNF)$, for any class $F$ of Boolean functions. The number of mistakes $A$ makes is bounded by $2^{k+1}n_k$.

**Proof:** Assume first that all the queries are $k$-disjunctions, and that the counterexamples are partial vectors in which exactly $k$ variables are not assigned $*$. It is clear that in this case we can make at most $2^k{n\choose k}$ mistakes (This amounts to learning the answer to any possible $k$-disjunction.). If the counterexamples have more than $k$ variables not set to $*$, it can only reduce the number of mistakes, since each counterexample can serve for testing more than one $k$-disjunction. Further, if the queries are $k$CNF instead of $k$-disjunctions the analysis does not change, since the counterexamples are enough to answer correctly all $k$-disjunctions, and therefore also their conjunctions. This holds for all the interpretations of partial assignments since we use the lazy reasoning algorithm. Finally observe that the number of counterexamples with less than $k$ known variables is at most $(2^0 + 2^1 + \ldots + 2^{k-1}){n\choose k} < 2^k{n\choose k}$ so that the overall mistake bound is guaranteed.  

We note that the mistake bound above is also a bound on the size of the set $\Gamma_f^k$, and that it is independent of the class $F$. That is, the algorithm makes no assumptions on the “world”, and its correctness is guaranteed provided only that the queries presented are restricted as stated. Also note that if the queries that $RQ$ presents are restricted to be $k$-disjunctions then the algorithm can find the counterexamples itself (simply set all literals in the disjunction to 0 and all others to $*$), and in this case we could have used a weaker oracle, that does not even supply the agent with counterexamples.

### 6.1.2 Conclusions

We have studied in this section an extension of the Learning to Reason framework to the study of deduction in the case where the interface to the world consists
of partially specified assignments. The intention was to argue that the problem of considering more general (read: reasonable) interfaces is important, and some interesting positive results can be achieved. The “Learning to Reason without Reasoning” result with \( k \)-CNF queries described above is just one interesting example. Other, positive and negative results in this line are described in (Khardon and Roth, 1994c) and we will not describe these here.

The rest of this chapter is devoted to the development of a more general approach, in which we no longer view the world \( W \) as a function ranging over \( \{0, 1\}^n \), and not necessarily even as a consistent function. The need to extend the discussion to this case arises from our inability to capture the non-monotonicity of reasoning with the approach suggested so far. In particular, we want to investigate the view that incomplete information may actually help in supporting plausible reasoning efficiently. In Section 6.3.1 we will compare the new approach which we discuss in the majority of this chapter to the one we have considered in this section.

### 6.2 Handling Incomplete Information: The Framework

We consider a set \( X = \{x_1, \ldots, x_n\} \) of variables, each of which is associated with a world’s attribute and can take the values 1 or 0 to indicate whether the associated attribute is true or false in the world. An agent interacts with the world through a set of \( d \) observed attributes \( v = (x_{i_1} = v_{i_1}, x_{i_2} = v_{i_2}, \ldots, x_{i_d} = v_{i_d}) \). (Throughout that chapter we use \( x_i \) to denote attributes, \( v_i \) to denote the corresponding values, and \( v \) to denote a vector over \( \{0, 1, \ldots, \}^n \).) Many of the unobserved attributes might not be known to the agent and the assignment to those, and to known attributes that are unobserved is denoted by the special symbol, \( * \). In this way, observations are vectors in \( \{0, 1, \ldots, \}^n \), but we write them by only specifying the observed variables.

The world \( W \) imposes some distribution \( D \) over \( \{0, 1, \}^n \) that governs the occurrences of the observations \( v \in \{0, 1, \}^n \) the agent sees. In general, we assume nothing about the world \( W \), nor about \( D \). Presumably, there are some functional dependencies in \( W \), e.g., \( x_1 = x_2 \wedge x_3 \), and those are respected by \( D \), in the sense that in any observation \( v \) drawn according to \( D \) if \( v_2 = v_3 = 1 \) then \( v_1 \neq 0 \).

In this framework the intelligent agent is given access to her favorite learning interface; by interaction with the environment she inductively learns her representation of the world and later uses it to respond to queries. Her performance on the reasoning task is measured in a way that makes explicit the dependence of the reasoning performance on the input from the world. In the next subsections we define the knowledge representation we investigate in this chapter, the type of interactions with the environment the agent might have and the reasoning tasks.

---

\(^3\)For example, the agent might not know of the attribute has_broken_wing.
Knowledge Representation

While we assume nothing about the world \( W \) or the distribution \( D \) it imposes on the observations the agent makes, we do need to specify the type of representation the agent uses to represent her knowledge about the world and how those can be manipulated while interacting with the world, when learning and reasoning.

We assume that for every known attribute \( x_j \) the agent maintains an attribute function \( f_j : \{0,1,*\}^{n-1} \to \{0,1\} \) that defines\(^4\) the dependence of \( x_j \) on the other attributes.

An attribute function \( f_j \) is represented in a way similar to the way we represent Boolean functions over \( \{0,1\}^n \), only that the set of values assigned to each attribute is a (non-empty) subset of \( \{0,1,*\} \) rather than a (non-empty) subset of \( \{0,1\} \) as is usually the case. For example, a conjunction \( f \) that depends on the attributes \( x_1, x_2, x_3 \) can be written as \( f = (x_1 = 1) \land (x_2 = 0 \text{ or } *) \land (x_3 = * \text{ or } 1) \). A DNF\(^5\) representation for \( f \) is written as

\[
f = \bigvee_{i=1}^{|s|} [(x_{i_1} \in s_{i_1}) \land (x_{i_2} \in s_{i_2}) \land \cdots (x_{i_k} \in s_{i_k})]
\]

where \( s_k \subseteq \{0,1,*\} \). A CNF representation is written in a dual manner. It is clear that using this notation every Boolean function over \( \{0,1,*\}^n \) can be represented as a DNF and as a CNF. Given a vector \( v \in \{0,1,*\}^n \) it is also easy to evaluate \( f(v) \).

We note that one can define also attribute functions as \( f_j : \{0,1,*\}^{n-1} \to \{0,1,*\} \), with the intended meaning that given \( v \in \{0,1,*\}^n \) in which the attribute \( x_j \) is unobserved, \( f_j(v) = * \) when the value of this attribute is undetermined. In this case \( f_j \) can be defined, for example, as a rooted tree whose internal nodes are attributes, and whose leaves are in \( \{0,1,*\} \). Each node in the tree has three children, corresponding to the three values of the label. Given an assignment, the function is evaluated recursively: starting at the root, we evaluate the \( i \)th subtree of the node if the node is labeled by an attribute that is assigned \( i \). It can be shown that the results we prove later in this chapter for functions which range over \( \{0,1\} \) hold also for functions which range over \( \{0,1,*\} \), but we will not be concerned with those in this chapter.

One of the main advantages the attribute-function knowledge representation has, is that we do not need to make any assumptions about the “world”. In particular, we do not need to assume that there exists a consistent world \( W \). In Section 6.5 we

---

\(^4\)In general, we might want to consider the case where \( x_j \) depends probabilistically on the other attributes, possibly as a p-concept (Kearns and Schapire, 1990). This is more reminiscent of the knowledge representation discussed in (Valiant, 1994a). In this chapter we restrict the discussion to deterministic representations.

\(^5\)We note that this look somewhat similar, but is different than the work on Boolean dependencies in Database theory. The subtle differences in the definitions result in different theories. See (Khardon, Mannila, and Roth, 1994) for a discussion of Boolean dependencies in Databases.
discuss other knowledge representations and reasoning tasks associated with them, and compare them to the attribute-function representation.

**Interaction**

We use oracles to model the type of interaction the agent has with the world, in the spirit of the formal study of learning and the general Learning to Reason framework. The oracles differ according to the amount and type of information they supply the agent about the world. In the following we define the oracles we use in this chapter since they are somewhat different that those we have used earlier. For the purpose of this exposition we assume that all the interactions of the agent with the world are done via observations \( v = (v_1, v_2, \ldots, v_d) \), possibly augmented by some other information.

**Definition 6.2.1** An Example Oracle with respect to the probability distribution \( D \) on \( \{0, 1, *\}^n \), denoted \( EX(D) \), is an oracle that when accessed, returns \( v \in \{0, 1, *\}^n \), where \( v \) is drawn at random according to \( D \).

We view this oracle as the main avenue of interaction with the world: the type of interaction which occurs in random situations. As discussed in Chapter 4 (and, more explicitly, in (Khardon and Roth, 1994a)), in situations constrained to satisfy some context condition (e.g., \( Q = \{x_1 = \text{we are in Boston}\} \) or \( Q = \{x_1 \land x_2 \rightarrow x_3\} \)), the occurrences of observations is not governed by \( D \) any more, but by some other distribution \( D_Q \) which is the distribution we see by filtering out all those observations that do not satisfy \( Q \). (We follow here the formulation suggested in (Valiant, 1994b)). We denote this oracle by \( EX(D_Q) \).

**Definition 6.2.2** A Membership Query Oracle for the attribute function \( f_j \), denoted \( MQ(f_j) \), is an oracle that when given an input \( v \in \{0, 1, *\}^{n-1} \), in which \( x_j \) is hidden, returns \( f_j(v) \). We denote the query by \( mq(v, j = \text{?}) \). An oracle that can answer membership queries for a class \( F \) of attribute functions is denoted by \( MQ(F) \).

**Definition 6.2.3** An Equivalence Query Oracle for the attribute function \( f_j \), denoted \( EQ(f_j) \), is an oracle that when given as input a function \( g : \{0, 1, *\}^{n-1} \rightarrow \{0, 1\} \), answers “yes” if and only if \( f_j \equiv g \). If it answers “no” it supplies a counterexample, namely, an \( v \in \{0, 1, *\}^n \) such that \( f_j(v) \neq g(v) \).

**Definition 6.2.4** A Causal Example Oracle with respect to the probability distribution \( D \) on \( \{0, 1, *\}^n \), denoted \( CEX(D) \), is an oracle that when accessed, returns \( (v, j) \) where \( v \in \{0, 1, *\}^n \) is drawn at random according to \( D \), and \( j \) is an index of an observed attribute in \( v \) that causally depends on the other attributes.

While receiving an observation from the example oracle \( EX(D) \) indicates some statistical correlation between the observed values and can thus be used by the
agent to update her representation for all the attribute functions of the observed attributes, the causal example oracle provides the agent additional information: $x_j$ causally depends on (a subset of) the observed attributes. Therefore, this example can be used to modify the representation of the attribute function $f_j$, but might not be useful for updating any of the other attribute functions. In that sense this oracle is stronger than the example oracle. While we assume here that the causality information is supplied to the agent, which we believe to be the case in many circumstances, it is quite possible, though, that in some cases one could use calls to an example oracle to simulate the causal example oracle. Much of the work on detecting causality (e.g., (Spirtes, Glymour, and Scheines, 1993)) is concerned with this issue.

The following oracle can be thought of as an on-line version of the example oracle:

**Definition 6.2.5** A Reasoning Query Oracle for the attribute function $f_j$ with respect to the distribution $D$, denoted $RQ_D(f_j)$, is an oracle that when accessed performs the following protocol with the agent $A$. (1) The oracle picks $v \in \{0,1,*\}^n$ according to $D$, hides the value of $x_j$ and returns it to $A$. (We denote the query by $rq(v, j = ?)$. (2) The agent $A$ answers “1” or “0” by evaluating $f_j(v)$. (3) The oracle responds by “correct” or “incorrect”. A reasoning query oracle for a class $\mathcal{F}$ of attribute functions is denoted by $RQ_D(\mathcal{F})$.

We denote by $I$ the interface available to the agent in a given situation. This can be any subset of the oracles defined above, and might depend on the arbitrary and unknown distribution $D$ over $\{0,1,*\}^n$ or some restriction of it, $D_Q$.

**The Learning to Reason Task**

The learning scenario most appropriate in our case is an on-line scenario (or, continuous learning) (Littlestone, 1989: Valiant, 1994a). In this scenario the algorithm is allowed access to the interface $I$. Every example received by the algorithm can be used to update many attribute functions in parallel. If $v \in \{0,1,*\}^n$ is supplied by $EX(D)$, and $v_j = 1, v_i = 0$ than $v$ can be used as a positive example for the attribute function $f_j$ and a negative example for $f_i$.

The reasoning task we consider is a *prediction* task. Given $v \in \{0,1,*\}^{n-1}$ in which $x_j$ is hidden (i.e., we do not receive a value for $x_j$), the algorithm is required to respond with $f_j(v)$. Thus, reasoning with respect to an attribute $x_j$ is reduced to evaluating the attribute function $f_j$ on a total vector over $\{0,1,*\}^{n-1}$. Reasoning in this case is just a classification task, and it depends on learning the correct attribute function.

For a class $\mathcal{F}$ of attribute functions we say that an algorithm solves the reasoning problem $RQ(\mathcal{F})$ if it can answer prediction queries with respect to all attribute functions $f \in \mathcal{F}$.

We consider a query $q$ given to the algorithm as if given by the reasoning oracle $RQ_D(f_j)$ (That is, $rq(v \in D, j = ?)$). Thus, a reasoning error supplies the algorithm
information which in turn can be used to improve its future reasoning behavior. In doing so, the algorithm may use other oracles from \( I \). Notice that the queries depend on the distribution \( D \), and thus the algorithm improves its performance “faster” in areas of the distribution in which it is queried more.

While in general we do not want to make assumptions about the world \( W \), in order to analyze the performance of our algorithms we need to assume something about the class of functions we work with. At this point we will make the assumption\(^6\) that the attribute functions are taken from some class \( \mathcal{F} \) of Boolean functions with domain \( \{0, 1, *\}^{n-1} \). (Notice that this does not even imply that \( W \) is consistent.)

As performance criteria we will use the criteria accepted in computational learning theory, namely, either the pac criterion (Valiant, 1984) or the mistake-bound criterion (Littlestone, 1989). For reasoning with attribute functions we can therefore define:

**Definition 6.2.6** An algorithm \( A \) is a Probably Approximately Correct Learning to Reason (PAC-L2R) (Mistake-Bound Learning to Reason (MB-L2R)) algorithm for the reasoning problem \( RQ(\mathcal{F}) \), if there exists a PAC (Mistake-Bound) learning algorithm for the class \( \mathcal{F} \), given access to \( I \).

We say that the algorithm is noise tolerant when it can tolerate the standard amount of classification noise\(^7\).

### 6.3 Default Reasoning

The term default reasoning is used in AI for patterns of inference that permit drawing conclusions suggested, but not entailed, by the knowledge available to the system. This is usually done by augmenting the available knowledge about the world with a set of default assumptions (or simply, defaults) that capture what is “typically” the case. The quest is for a reasoning system that, given a query, responds in a way that agrees with what we know about the world and some subset of the default assumptions and at the same time supports our intuition about a plausible conclusion.

Attempts to represent and reason with defaults in AI have encountered many problems. In many cases, reasoning with “acceptable” defaults leads to unacceptable conclusions. Problems occur whenever default interact, and in most cases they can be characterized as problems of distinguishing “good” defaults from “bad” ones.

\(^6\)It is possible that a better framework to discuss here is that of agnostic learning (Kearns, Schapire, and Sellie, 1992). However, very few positive results are known in this framework and, as we will see later, it seems as if our learning problems tend not to require an expressive class of functions anyhow.

\(^7\)Classification noise occurs when there is some probability \( \eta \) (the error rate) that the label of an example is flipped (from 0 to 1 or vice versa). Most learning algorithms known can tolerate classification noise with error rate \( \eta < 1/2 \).
But reasons for deciding which defaults are good and which are bad vary, and in most cases depend on the situation. Some authors have classified those as problems of "specificity", where one default rule takes into account more information; problems of "irrelevance", where a default rule is not relevant to the current query, and others. As argued in (Geffner, 1990; Pearl, 1988), no general method exists, according to which one can rank defaults. The only way to figure out why and when certain defaults are preferred to others is to understand what the defaults say about the world. While the probabilistic approach taken in (Geffner, 1990) presents an important step in this direction, it still suffers from some of the same problems (e.g., (Geffner, 1994)), and of course, is infeasible computationally.

Unlike previous theories of reasoning in the presence of incomplete information, we are not interested in providing a theory of defaults, but rather a theory of inference. Defaults do not exist in the world. One cannot evaluate whether a default is true in the world but only if a conclusion derived at by a reasoning system is acceptable.

As we show later, in Section 6.3.1, there is no direct mapping between the way default reasoning problems have been traditionally defined and our framework. In order to exhibit the advantages of our approach we translate default reasoning problems into Learning to Reason problems as follows: given a default reasoning problem (i.e., "true" world knowledge and a set of default assumptions) we suggest a scenario of interactions with the world that, we believe, reflects the type of observations that could have led to this view of the world. We use those observations to construct an attribute-function representation of the world, over \( \{0, 1, *\}^n \), and then argue that given a query, this representation yields the sought after response.

We disregard in this section the issue of how to compute the attribute function from the observations and the cost of this computation. One of the surprising outcomes of this translation exercise is that all the examples used in the literature have attribute functions with very simple structure, so it will be easy to construct those manually from the observations. This issue is further discussed in Section 6.4.1.

We use the following convention in the presentation of the default reasoning problems. The traditional representation is given as a set \( KB \) of knowledge base rules and a set \( D \) of default rules. (As usual, \( penguin(x) \rightarrow bird(x) \) means that if \( x \) is a penguin then \( x \) is a bird.) For each problem we present a set of observations about the world. Those observations are elements in \( \{0, 1, *\}^n \), but we present only the observed attributes, and only those which are of some interest to the current problem. As usual, all the unobserved attributes are assigned \( * \).

All the examples discussed below have been studied before in the literature. The examples, or versions of them, represent various aspects of the non-monotonic reasoning phenomena that have been used over the years as "bench-marks" for various formalisms. We do not know of any "traditional" formalism that can handle in a satisfying way (efficiently, or even qualitatively) all the aspects presented by those examples. We note, though, that our first example is a variant of an example considered in (Valiant, 1994a), and that all the examples we consider here could be
considered also in the *Rationality* framework (Valiant, 1994b) and be implemented in principle on the Neurooidal Model (Valiant, 1994a). A (partial) list of other papers that have discussed (a subset of) these examples includes (Bachus et al., 1993; Etherington, 1988; Geffner, 1990; Reiter, 1980; Reiter and G., 1981; Selman, 1990; Touretzky, Hory, and Thomason, 1987).

**Example 6.3.1 (Basic Example)** Consider the case in which we know that penguins are birds, penguins do not fly, and we also have the “default” assumption “birds fly”. This is expressed as the set of facts

\[ KB = \{ penguin(x) \rightarrow bird(x), penguin(x) \rightarrow \overline{fly}(x) \} \]

and the default statement

\[ D = \{ bird(x) \rightarrow fly(x) \}. \]

Given this, it is reasonable to assume that in all observations we made so far of the world, whenever we saw an example in which the `penguin` attribute was ‘on’ (set to 1) the `bird` attribute was 1 as well and the `fly` attribute was set to 0. Moreover, we have seen examples in which `bird` was 1 and `fly` was 1. In those examples, `penguin` was never 1. That is, a plausible sequence of examples could be:

\[
\begin{align*}
(bird = 1, penguin = 1, fly = 0) \\
(bird = 1, fly = 1) \\
(bird = 1, fly = 1, red = 1) \\
(bird = 1, fly = 1, red = 0) \\
(bird = 1, penguin = 0, fly = 1, has_beak = 1) \\
(bird = 1, fly = 1, has_beak = 1) \\
(bird = 1, penguin = 1, fly = 0, has_beak = 1)
\end{align*}
\]

Recall that we assume that all the attributes not mentioned explicitly in the examples have value `*`.

Given these observations the attribute function an agent would keep for `fly` is

\[ f_{fly} = (bird = 1) \land (penguin = 0 \text{ or } *). \]

Consider now a query regarding Tweety, \( rq((bird = 1), fly = ?) \). (This notation implies that the attributes not mentioned have value `*`.) In this case, all we know is that Tweety is a bird (that is, in this observation, the only observed attribute is `bird`), and evaluating \( f_{fly} \) yields the prediction `fly` = 1.

It is quite possible that the observations listed above are all the observations the agent had seen that had to do with birds. It is more likely, though (and is our intention in this section) that those examples are just a “representative” sample of what she had seen. In this case, along with seeing many examples similar to the above, the agent could also see a small number of examples like \( (bird = 1, fly = 0) \). We comment that this is still okay even with our deterministic representation of the attribute functions. We view those cases as *classification noise*, that is, cases where the value supplied by \( EX(D) \) for the function \( f_{fly} \) is false. In Section 6.4.1 we show that the learning algorithms suggested can tolerate a large amount of classification noise, and therefore we will not incorporate misclassified observations in the next examples.
Example 6.3.2 (Specificity) Consider the set of examples discussed in Example 6.3.1 and assume we are given a query about the penguin Tweety: \( q((\text{bird} = 1, \text{penguin} = 1), \text{fly} = ?) \). Clearly, in this case, evaluating \( f_{\text{fly}} \) yields the prediction \( \text{fly} = 0 \). That is, we conclude that Tweety does not fly, even though Tweety is a bird and birds (when no other, more specific, information is known) fly.

Example 6.3.3 (Irrelevance-I) Consider the set of examples discussed in Example 6.3.1 and assume we are given a query about the red bird Tweety. That is, the query regarding Tweety is presented as: \( q((\text{bird} = 1, \text{red} = 1), \text{fly} = ?) \). Clearly, the observations we made show that the attribute \( \text{red} \) is irrelevant to the attribute function \( f_{\text{fly}} \) and evaluating it therefore yields the prediction \( \text{fly}(\text{tweety}) = 1 \).

Of course, an agent active in a green birds natural reserve might “be trained” on a different set of observations and see green birds (almost) exclusively. As a result she might believe that “green-hood” is a necessary property of flying birds, that is, she might have \( f_{\text{fly}} \equiv (\text{bird} = 1) \land (\text{green} = 1) \) as the attribute function for fly. There is no contradiction here; this is exactly the type of reasoning patterns the sought after theory should possess.

Example 6.3.4 (Irrelevance-II) Consider the set of examples discussed in Example 6.3.1. In this case we are interested in predicting the attribute \( \text{has_beak} \). The query about the penguin Tweety is presented as: \( q((\text{bird} = 1, \text{penguin} = 1), \text{has_beak} = ?) \). Predicting in this case is done by evaluating the attribute function \( f_{\text{has_beak}} \). The important point to note here is that there is no relation between the attribute functions \( f_{\text{has_beak}} \) and \( f_{\text{fly}} \). Those are acquired in parallel and the fact that penguins have special properties with respect to flying does not mean they have to have exceptional properties with respect to “having a beak”. In this case the observed examples lead to \( f_{\text{has_beak}} \equiv (\text{bird} = 1) \), and evaluating it yields \( \text{has_beak} = 1 \).

We note that while the conclusion above is very intuitive, it is not supported by most treatments of default reasoning (e.g., (Makinson, 1989; Kraus, Lehmann, and Magidor, 1990)), which encounter difficulties in trying to support both specificity and irrelevance.

Example 6.3.5 (Multiple Extensions) Consider the set of facts
\[ KB = \{ \text{bat}(x) \rightarrow \text{mammal}(x) \} \] and default statements
\[ D = \{ \text{mammal}(x) \rightarrow \text{fly}(x), \text{bat}(x) \rightarrow \text{fly}(x), \text{dead}(x) \rightarrow \text{fly}(x) \} \]. It is then reasonable to assume that the observations we made so far of the world have the following properties: in examples with a bat attribute set to 1, the mammal attribute was 1 as well; we have seen examples of bats that fly but also examples of mammals that do not fly; in the latter case, bat was not 1; also, we have not seen dead things fly. Therefore, a plausible set of observations could be:

\( (\text{mammal} = 1, \text{bat} = 1, \text{fly} = 1) \)
\( (\text{bat} = 1, \text{fly} = 1) \)
\( (\text{mammal} = 1, \text{fly} = 0) \)
\( (\text{mammal} = 1, \text{bat} = 0, \text{fly} = 0, \text{red} = 1) \)
Given these observations the attribute function an agent would keep for fly is $f_{\text{fly}} = (\text{bat} = 1) \wedge (\text{dead} = 0 \lor \text{fly} = 0)$. Consider now a query regarding Dracula, presented as: rq((\text{bat} = 1, \text{dead} = 1), \text{fly} = ?). Clearly, evaluating $f_{\text{fly}}$ on this example yields the prediction $\text{fly}(\text{dracula}) = 0$. In case all we know is that Dracula is a bat and we do not know that it is dead (that is, dead = *) the query is $rq(\text{bat} = 1, \text{fly} = ?)$ and evaluating $f_{\text{fly}}$ this time yields the prediction $\text{fly}(\text{dracula}) = 1$.

As before, there is no contradiction here; this is exactly the type of reasoning pattern the sought after theory should possess.

The traditional treatment runs in this case into problems of conflicting defaults. For example, one has to decide which of the default rules, $bat(x) \rightarrow fly(x)$ or $\text{dead}(x) \rightarrow \text{fly}(x)$ to apply in order to predict the value of $\text{fly}$(Dracula), or, alternatively, to add some ad hoc rules in order to prefer one default over the other. Essentially, these ad hoc additions are done in order to capture the type of information given by the observations, and as we show, this information can be represented easily as an attribute function $f : \{0, 1, \ast\}^n \rightarrow \{0, 1\}$.

**Example 6.3.6 (Preferences)** Consider the case in which the default statements are given by $D = \{\text{student}(x) \rightarrow \text{employed}(x), \text{adult}(x) \rightarrow \text{employed}(x), \text{student}(x) \rightarrow \text{adult}(x)\}$ and the set of facts is empty. Those default were written in that way to reflect a situation in which the agent observes examples with the following properties: in examples in which the student attribute was set to 1 the employed attribute was not set to 1; in examples in which the student attribute was set to 1 the adult attribute was not set to 0; in examples in which the adult attribute was set to 1, the employed attribute was not set to 0, unless some other information is given.

The following sequence of example could have been observed by the agent:

- $(\text{student} = 1, \text{employed} = 0)$
- $(\text{student} = 1, \text{adult} = 1)$
- $(\text{employed} = 1, \text{adult} = 1)$
- $(\text{student} = 0, \text{employed} = 1, \text{adult} = 1)$
- $(\text{student} = 1, \text{employed} = 0, \text{adult} = 1)$

Given these observations the attribute function an agent would keep for employed is $f_{\text{employed}} = (\text{adult} = 1) \wedge (\text{student} = 0 \lor \text{fly} = 0)$. On the other hand, these observations do not give us enough information to support prediction of the attribute adult in a simple way.

---

8We could disjunct it with the attribute function from Example 6.3.1, but will assume, for clarity, that those are different agents.
We note that most non-monotonic logics in AI require the explicit addition of preferences in order to deal with interacting defaults (Reiter and G., 1981). In Section 6.3.1 we discuss model preferences approaches and compare them to our approach.

In some cases, not only is there more information relevant to predicting one attribute than to others, as above, but some of the information might even mislead the prediction of some attributes. This is discussed in the default reasoning literature as the issue of causality. The following example is taken from (Geffner, 1990).

**Example 6.3.7 (Causality)** We get up in the morning and want to drive to work. When we get to the car, however, we notice that we left the lights on the previous night. At this point we want to assess the chances that the car will start upon turning the ignition key. This situation is modeled by assuming an empty set of facts and a set of default statements given by

\[ D = \{ \text{turn_key}(x) \land \text{battery_dead} \rightarrow \text{starts}(x), \text{turn_key}(x) \rightarrow \text{starts}(x), \text{lights_were_on}(x) \rightarrow \text{battery_dead}(x) \}. \]

Those default were written in that way to reflect the assumptions that usually, the car starts when the ignition key is turned on; when the battery is dead the car does not start, and after leaving the lights on for a night, the battery is usually dead. The following observations could have led to those assumptions:

- \( \text{turn_key} = 1, \text{starts} = 1 \)
- \( \text{turn_key} = 1, \text{lights_were_on} = 1, \text{starts} = 0 \)
- \( \text{lights_were_on} = 1, \text{starts} = 0 \)
- \( \text{battery_dead} = 1, \text{starts} = 0 \)
- \( \text{turn_key} = 1, \text{battery_dead} = 1, \text{starts} = 0 \)
- \( \text{lights_were_on} = 1, \text{battery_dead} = 1 \)

Given these observations the attribute function an agent would keep for \( \text{starts} \) is \( f_{\text{starts}} \equiv (\text{turn_key} = 1) \land (\text{lights_were_on} = 0 \lor \ast) \land (\text{battery_dead} = 0 \lor \ast) \).

Therefore, given the query \( rq(\text{battery_dead} = 1, \text{starts} = ?) \) we predict that the car will not start. We cannot, however, use this information to predict that the battery is dead, given the query \( rq(\text{starts} = 0, \text{battery_dead} = ?) \), even though if we were to learn an attribute function for \( \text{battery_dead} \) from these observations, we could have used it to wrongly predict \( \text{battery_dead} = 1 \). The reason is that there is some causal asymmetry between attributes in this example. Unlike other cases, where we can use the given observations to learn an attribute function for each attribute in terms of the other attributes, in this case some additional causality information might be required along with the observations informing the agent to only learn \( \text{starts} \) in terms of other attributes.

\(^9\)We deliberately conceal the temporal aspects of this example.

\(^{10}\)There are other ways to explain the problem in this example; e.g., as an abductive process. While we do not discuss it here, it is also supported by this theory (see Chapter 4 and Section 6.5).
Example 6.3.8 (Frame Problem) There are, of course, many other reasons for the car in Example 6.3.7 not to start, (e.g., someone could cut_the_wires leading from the ignition to the battery). This is related to the frame problem in AI, which is concerned with how to indicate which aspects of the world do not change when an action takes place (McCarthy and Hayes, 1969). While the standard non-monotonic reasoning formalisms do not capture the desirable behavior, that things stay as they are (Hanks and McDermott, 1986), our representation of incomplete information does. If we don't know of the existence of the attribute cut_the_wires it will not be part of the attribute function representation for starts. Moreover, even when we know of the attribute battery_dead = 1, if we do not observe it, and just present the query \( \text{rq(turn_key = 1, starts = ?)} \) (which means, lights_were_on = *, battery_dead = *) the response is, as it should be, starts = 1.

What is most striking about the examples presented above is not only the fact that all the examples, with which various default reasoning formalisms struggle, have a unified representation in our framework, but that further:

Observation 6.3.1 In all the cases presented above the attribute function for the attribute of interest can be represented as a conjunction.

It is an empirical question whether there are naturally arising reasoning problems in which the sought after attribute cannot be represented as a simple function over \( \{0, 1, *\}^n \). As we show in Section 6.4, since in these simple reasoning tasks, reasoning reduces to function evaluation, we can actually learn to reason with function classes which are a lot richer than is needed in the examples discussed above.

6.3.1 Relation to Default Reasoning Formalisms

Many formalisms for default reasoning have been shown to be computationally hard (e.g., (Selman, 1990; Papadimitriou, 1991; Kolaitis and Papadimitriou, 1988)). It is therefore interesting to consider how our approach compares to those studies. Consider for example the approach to default reasoning that is based on preferred interpretations. In this approach a theory \( \Phi \) and a set \( D \) of defaults are given. The theory defines a set of possible models of the world, and the default rules in \( D \) define a subset of preferred models. There are various ways of formalizing a preference ordering among models. One well known method used is circumscription, in which minimal models are preferred (McCarthy, 1980; McCarthy, 1986) (A propositional version of circumscription is discussed in (Papadimitriou, 1991)). Another method to define preferred models is discussed in (Selman and Kautz, 1990; Papadimitriou, 1991). In this approach the goal is to find a model of the theory that is maximal with respect to a partial order defined on the models by the default rules. In the preferred models formalisms, once a models is found, inference is done by evaluating a given query in this model. While this formalism leads to some intriguing mathematical problems, we argue that one need not solve these problems in order to reason in a way that agrees with the incomplete “default” information. First, the Learning to
Reason approach avoids the satisfiability problem that is hidden in these formalisms of default reasoning. As discussed in Chapter 3, this is done by having direct access to the world, and thus side-stepping the difficulties introduced by using representations that hide the models of the world. Second, in our approach we do not look for some “global” minimal (or maximal) model, but only for a simple explanation that agrees with the data.

Consider, for example, the case discussed in Example 6.3.6. There, no minimal model exists that can capture the “intuitive” inference with respect to all the attributes. Given the observations, the attribute function the agent would keep for employed is \( f_{\text{employed}} = (\text{adult} = 1) \land (\text{student} = 0 \lor \star) \). On the other hand, these observations do not give us enough information to support prediction of the attribute adult in a simple way. The observations support the following DNF-like attribute function for adult: \( f_{\text{adult}} = ((\text{employed} = 1) \land (\text{student} = 0 \lor \star)) \lor ((\text{employed} = 0 \lor \star) \land (\text{student} = 1)) \). It seems therefore as if trying to use a single model in \( \{0,1\}^n \) to characterize the situation not only makes the problem harder computationally, but does not even support the “intuitive” conclusion. Our approach, on the other hand, uses the data available to characterize the situations in which a specific attribute is “on”. This can always be done, and the only question remains is how complex is the representation of this characterization.

### 6.4 Learning to Reason

Reasoning with respect to an attribute \( x_j \) is reduced in this framework to evaluating the attribute function \( f_j \) on a total vector in \( \{0,1,\star\}^{n-1} \). Reasoning in this case is essentially a classification task, and depends on learning the correct attribute function. Definition 6.2.6 indicates that whenever we have a learning algorithm for a class \( \mathcal{F} \) of Boolean functions over \( \{0,1,\star\}^n \), we have a Learning to Reason algorithm for the corresponding class of attribute functions.

It turns out that most of the existing learning algorithms for Boolean functions studied in computational learning theory (see, e.g., (Blum et al., 1994), for a survey) can be extended with little effort to learning algorithms over \( \{0,1,\star\}^n \). We discuss next the learnability of attribute functions for the case of learning Boolean function with examples only and learning with membership queries.

#### 6.4.1 Learning to Classify over \( \{0,1,\star\}^n \)

**Learning from Examples Only**

Given the extensive existing literature on learning Boolean functions from examples (see, e.g., (Kushilevitz and Roth, 1993)) it is sufficient for our purposes here to argue that those algorithms apply also for the learnability of functions \( f : \{0,1,\star\}^n \rightarrow \{0,1\} \).

Consider for example the standard elimination algorithm for learning conjunctions (Valiant, 1984). It is easy to see that the same algorithm works even if the
values assigned to the variables are non-empty subsets of \{0, 1, *\} rather than non-empty subsets of \{0, 1\} as is usually the case. In the usual elimination algorithm the convention used is that when a variable \(x_i\) is allowed to have any value in \{0, 1\}, we omit it from the conjunctive representation. We use the same convention here. Moreover, we use the same convention for variables that have never been observed. In order for variables that were not observed yet (i.e., never appeared as 0 or 1) not to appear in the conjunctive representation, the algorithm uses the first positive example to initialize its hypothesis. From then on, it (1) adds to the conjunction only newly observed attributes, and (2) uses elimination over the set of known attributes, as is done in the usual elimination algorithm.

Using the techniques\(^\text{11}\) introduced in (Kushilevitz and Roth, 1993) we can show how to learn \(k\)DNF and \(k\)CNF formulae over \{0, 1, *\}\(^n\), for any fixed \(k\). For example, to see that \(k\)DNF is learnable it is enough to observe that given a positive example \(v \in \{0, 1, *\}^n\) and a set of \(k\) attributes, there are \(3^k\) possible terms that use only these attributes and are satisfied by \(v\) (since there are 3 non trivial subsets of \{0, 1, *\} that contain an assigned value). A similar argument shows that \(k\)CNF is learnable. As shown in (Kushilevitz and Roth, 1993) all the function classes that are polynomially explainable are learnable in the presence of various types of noise, to support a desirable property mentioned earlier. (See the discussion after Example 6.3.1.) To summarize.

**Theorem 6.4.1** Let \(\mathcal{F}\) be the class of all conjunctions, disjunctions, \(k\)CNF and \(k\)DNF formulae over \{0, 1, *\}\(^n\). Then, there exists an efficient and noise tolerant PAC-L2R (MB-L2R, respectively) algorithm for the reasoning problem \(RQ(\mathcal{F})\), that uses the example oracle \(EX(D)\) \((RQ_D(f_j),\) respectively\).

**Proof:** Assume that \(f_j \in \mathcal{F}\) is the attribute function for \(x_j\), and let \(A\) be a pac learning algorithm for the class \(\mathcal{F}\). A PAC-L2R algorithm uses calls to \(EX(D)\). Given \(v \in \{0, 1, *\}^n\) in which the attribute \(x_j\) is observed, it passes a labeled example \((v^{(j)}, v_j)\) to \(A\). Here, \(v^{(j)} \in \{0, 1, *\}^{n-1}\) is the total vector defined by \(v_i^{(j)} = v_i\) for \(i < j\) and \(v_i^{(j)} = v_{i+1}\) for \(i \geq j\) (as usual, with * for all the unobserved attributes), and \(v_j\) is the label, positive example when \(v_j = 1\) and negative example when \(v_j = 0\). The output of algorithm \(A\) is then used to predict the value of \(x_j\), given a prediction query. A similar procedure is run for the Mistake Bound algorithm, only that the oracle \(RQ\) is used in this case.

**Using Membership Queries**

It is well known (Angluin, 1988; Blum et al., 1994; Bshouty, 1993a) that richer classes of Boolean functions can be learned using membership queries, and in this section we discuss the generalization of those algorithms to functions which range

---

\(^{11}\)The notion introduced there is that of polynomial explainability. Informally, if given a positive example of \(f\), there exists an efficient algorithm that enumerates polynomially many terms, at least one of which is in the DNF representation of \(f\), then \(f\) is learnable.
over \( \{0, 1, *\}^n \). First, we describe again the notion of a membership query in this case. Given \( v = (x_{i1} = v_{i1}, x_{i2} = v_{i2}, \ldots, x_{id} = v_{id}) \in \{0, 1, *\}^n \) in which the attribute \( x_j \) is hidden, a call \( mq(v, j = ?) \) to the membership query oracle, is a request for the value of \( f_j \), under the assumption that all the attributes that were not assigned values in \( v \) are unobserved, that is, are equal to *. When using membership queries to learn Boolean function over \( \{0, 1\} \), all the known learning algorithms make use of the natural partial order on \( \{0, 1\} \). It seems as if this is unavoidable also in the case of functions defined over \( \{0, 1, *\}^n \).

In (Bshouty, 1993b) it is shown how to extend the monotone theory (Bshouty, 1993a) to hold for functions defined over a larger alphabet. To do that one must impose an algebraic structure on \( \{0, 1, *\} \), and define a partial order on it. \( \{0, 1, *\} \) can be made a group\(^{12}\) with 0 being the identity element and by defining \( 1 + 1 = *, * + * = 1, * + 1 = 0 \). It is shown in (Bshouty, 1993b) that for the monotone theory to hold the partial order defined on \( \{0, 1, *\} \) must have one minimal element and two maximal elements. Thus the 0th partial order is: \( 0 <_0 1, 0 <_0 *, * \) and 1 are incomparable. From it, using the group structure defined we get two other order relations on \( \{0, 1, *\} \). \( * <_* 1, * <_* 0, 0 \) and 1 are incomparable, and \( 1 <_1 *, 1 <_1 0, 0 \) and * are incomparable.

It is then shown that the results of the monotone theory hold in this case as well. Phrasing those results as a Learning to Reason results we have:

**Theorem 6.4.2** There exists an efficient PAC-L2R algorithm that uses \( RQ_D(f_j) \) and \( MQ(f_j) \), for the reasoning problem \( RQ(F) \) where

(i) \( F \) is the class of Decision Trees over \( \{0, 1, *\}^n \).
(ii) \( F \) is the class of log \( nCNF \cap DNF \) over \( \{0, 1, *\}^n \).

We note that the representations over \( \{0, 1, *\}^n \) supported by the monotone theory are more restricted than those we have allowed in Section 6.2. For example, a conjunction must be of the form \( \bigwedge_{i=1}^m (x_i = v_i) \), for \( v_i \in \{0, 1, *\} \), and accordingly, a DNF representation is restricted to contain only conjunctions of this form. Of course, every Boolean function over \( \{0, 1, *\}^n \) can be represented in this way (as a DNF) but this representation will be larger than a representation of the form defined in Section 6.2. Since the complexity of the algorithms depend on the size of the representation, smaller representations are preferred, and it will be interesting to extend those algorithms to hold for the more general representations.

### 6.5 General Reasoning Tasks

#### 6.5.1 Answering General Queries

So far in this chapter we have discussed prediction queries, that is, queries with respect to a single attribute. While this is a fairly general reasoning task, and the

\(^{12}\text{There is a unique group of order } 3, \{\{0, 1, 2\}, + \text{ (mod 3)}\}.\)
one considered in most of the works on default reasoning, the general literature on inference considers also more general queries and we will consider those next.

In general, the reasoning task facing an agent is given as a query, a formula \( \alpha \), that captures the situation at hand. Given a description \( W \) of the world, possibly augmented by a set \( D \) of defaults, the agent is supposed to decide whether \( \alpha \) holds. (We call this below the general reasoning task.)

Clearly, this formulation of the reasoning task is more general than the one considered earlier, since (as we show below) every \( rq \) query can be written as the \( \alpha \) query above. Nevertheless, there are few reasons that justify using the attribute-function representation in this work aside from the fact that it is easier to compare this reasoning task with the tasks commonly studied in default reasoning (i.e., queries with respect to a single attribute). First, prediction tasks seem to be fairly general; combining and chaining prediction tasks might be sufficient to deal with reasoning in the temporal domain (Shoham, 1988) and to reason about action. Second, the general reasoning task implies some uniformity in the representation, in the sense that we can, in principle, answer queries with respect to all attributes. While the Learning to Reason framework remedies this and supports a bottom-up approach to reasoning (as in Section 4.7), this is made more explicit when using the attribute-function representation. Here, depending on the agent’s observations, she could answer queries with respect to some of the attributes, but possibly, not all. Third, reasoning with attribute functions is always computationally easy, as it reduces to simple function evaluation.

Notice that having an attribute-function representation does not imply that there is a consistent function \( W \) that agrees with all the attribute functions. In this subsection, however, for the sake of comparing the \( rq \) queries with queries on \( W \), we assume that there is a function \( W : \{0,1,*\}^n \rightarrow \{0,1\} \) that agrees with them. We are interested in the relations between answering queries with respect to \( W \) and answering queries using the attribute-function representation. (Recall that we can view \( W \) as if already augmented by a set \( D \) of defaults, for the purpose of this comparison, and the agent is supposed to decide whether \( \alpha \) holds in \( W \).)

Assume that the situation observed in the world is described by assigning values to the attributes \( \{x_i\}_{i \in I} \), and we are interested in the attribute \( x_j, (j \notin I) \). The “general reasoning task” in this case is phrased as follows: Let \( m_\gamma = \bigwedge_{i \in I} (x_i = v_i) \) be a monomial and consider the query \( \alpha_j \equiv m_\gamma \rightarrow x_j \). We say that \( \alpha_j \) holds in \( W \) (denoted \( W \models \alpha_j \)) if and only if all the models of \( W \) satisfy \( \alpha_j \). That is, it holds regardless of the values assigned to attributes \( x_k, \) for \( k \notin I \).

In this chapter, when the situation observed in the world is described by assigning values to the attributes \( \{x_i\}_{i \in I} \), the reasoning query considered is \( rq(v, j =?) \), where \( v = (x_i = v_i)_{i \in I} \). That is, we assume that all the other attributes are unobserved. The query \( rq \) can therefore be re-written as follows: Define \( m_\ast = \bigwedge_{i \in I} (x_i = v_i) \land \bigwedge_{i \notin I, i \neq j} (x_i = *) \). Clearly, since \( m_\ast \) holds in the world \( W \), predicting \( rq(v, j =?) = 1 \) is equivalent to answering ‘yes’ to the query

\[
W \models (m_\ast \rightarrow (x_j = 1)).
\] (6.1)
That is, stating the $rq$ queries as a general reasoning task reveals that we answer queries of the form $m_\ast \rightarrow (x_j = 1)$. Answering the $m_\ast \rightarrow (x_j = 1)$ query is easier than answering the $m_{W} \rightarrow (x_j = 1)$ query, since it reduces to a simple function evaluation, but clearly, in general the answer is different.

When the answer to the query $m_\ast \rightarrow (x_j = 1)$ is negative, that is, when $f_j(v) = 0$, the answer to $m_{W} \rightarrow (x_j = 1)$ is also negative. Assume now that $m_{W} \models f_j$. In this case, $f_j(v) = 1$ clearly implies (not only that $m_\ast \rightarrow (x_j = 1)$ holds is $W$ but also) that $m_{W} \rightarrow (x_j = 1)$ holds is $W$. We have:

**Claim 6.5.1** *Evaluating the attribute function $f_j$ is equivalent to answering the query $m_{W} \rightarrow (x_j = 1)$ when $m_{W} \models f_j$.***

We call the condition stated in the claim above, in which the situation observed in the world, when asked about the attribute $x_j$, determines $f_j$, the case of *simple queries*. This condition trivially holds when $f_j$ depends only on variables that appear in $m_{W}$, that is, on the attributes observed in the current situation.

Consider now the query $\alpha_{ij} = m_{W} \rightarrow (x_j \lor x_i)$. As before, it is easy to see that if $m_{W} \models f_j$ and $m_{W} \models f_i$ (and in particular, if $f_i$ and $f_j$ depend only on variables that are assigned values by $m_{W}$) then it is sufficient to evaluate $f_i$ and $f_j$ on $v$, the total vector corresponding to $m_{W}$, and

$$W \models \alpha_{ij} \equiv W \models (m_{W} \rightarrow (x_j \lor x_i)) \equiv f_j(v) \lor f_i(v).$$

Now, consider the case of a compound query, $\alpha_i \land \alpha_j = (m^{(i)} \rightarrow (x_i \lor \ldots x_{i_n})) \land (m^{(j)} \rightarrow (x_j \lor \ldots x_{j_m}))$, where $m^{(i)}$, $m^{(j)}$ are monomials. Since $W \models \alpha_i \land \alpha_j$ if and only if $W \models \alpha_j$ and $W \models \alpha_i$, the correctness of answering the queries using the attribute-function representation depends on its correctness on the conjuncts queries. To summarize, we have that

**Claim 6.5.2** *All simple queries can be answered by evaluating the appropriate attribute functions.*

In general, however, when the queries are not “simple”, evaluating the corresponding attribute functions does not yield the same answer. Notice that in this case the “mistake” is one sided: attribute-function evaluation is “correct” when it answers ‘no’.

The discussion above shows that even when we do not have the attribute functions representation, but rather a representation of $W$ (not necessarily exact, as we show in the next section), we can still reason according to the interpretation we suggest here: given a situation observed in the world, we should use $m_{\ast}$ rather than $m_{W}$ when we translate the observed situation into a query. The question of whether compound queries are needed at all, and if the $m_{W}$ interpretation is valid in certain situations, is an empirical question. In Section 6.3, we have shown that the attribute-function (or, the “$m_{\ast}$”) interpretation supports the sought after conclusions in scenarios of reasoning with incomplete information. We view it as evidence that this is indeed the right interpretation for these reasoning tasks.
6.5.2 Relation to Model-Based Reasoning

Learning the attribute-function representation and reasoning with it, aside from being efficient, has the advantage that it does not force us to make any assumptions about the world $W$, and in particular, we do not need to assume that $W$ is consistent. On the other hand, as discussed above, given a query $\alpha$, assumed to capture the situation at hand, reasoning with the attribute-function representation takes a specific view on the attributes that are unobserved in this situation. We have argued before that this is the correct interpretation for supporting plausible inference.

The approach suggested in this chapter relies on (1) interaction with the world in order to learn and maintain a knowledge representation, and (2) using a generalized representation over $\{0, 1, *\}^n$ that supports reasoning with incomplete information. But, it does not necessitate using the attribute functions as a sole representation. In particular, we could adopt the interpretation of incomplete information taken here, and combine it with the approach developed earlier in this thesis: learn a representation for $W$ (or, better, the state of “truth” in $W$) and reason with it. Notice that, as shown in Chapter 4, we can trade making assumptions on $W$, by making assumptions on the class of queries. If we assume that all the queries are from some class $\mathcal{Q}$ of Boolean functions (over $\{0, 1, *\}^n$), it is sufficient to learn the least upper bound of $W$ with respect to $\mathcal{Q}$ in order to support correct reasoning.

Given that $\text{KB}=W_{\text{tab}}$ is learned as a Boolean function over $\{0, 1, *\}^n$, reasoning with it, with simple queries in $\mathcal{Q}$, yields the same answers as using the attribute functions. (For non-simple queries, we need to translate the situation observed in the world according to the $m_\ast$ interpretation presented in the previous section, in order to agree with the attribute-function interpretation.) In particular, in all the examples discussed in Section 6.3, reasoning still yields the sought after conclusion, as shown there. Moreover, all the results discussed earlier in the thesis transfer to the $\{0, 1, *\}^n$ domain, as discussed in Section 6.4. In particular, one can construct a model based representation over $\{0, 1, *\}^n$ and reason with it, and, as shown in Section 4.7 one can combine the model-based representation with a set of restricted “context” rules, and still reason correctly.

6.5.3 Deduction

All theories of default reasoning studied in the knowledge-based system paradigm share the desirable property that they reduce to deductive reasoning when the knowledge base consists only of rules and facts about the world, and there are no defaults. This property trivially holds in the framework suggested here. Having complete information means here that all the observations are total vectors $v \in \{0, 1\}^n$, and the functions we learn range over $\{0, 1\}^n$, as in the theory of Learning to Reason for deduction, presented in Chapter 5.
6.5.4 Related Work

We have already discussed the relation of our work to some of the vast literature on default reasoning and therefore concentrate here on works that are related to ours in the approach to reasoning.

In Section 6.1 we briefly discussed an extension of the Learning to Reason framework to the case in which the interaction of the learner with the world supplies only partial information. There, we concentrated on deductive reasoning, and a basic premise was that the world can be represented as a Boolean function over \( \{0, 1\}^n \). Various ways were considered there in which to interpret what a partial observation conveys about the world.

Given a function \( g : \{0, 1\}^n \rightarrow \{0, 1\} \), an \textit{interpretation} defines an extension of the function \( g \) to the \( \{0, 1, *\}^n \) domain. Therefore, it is clear that for any function \( g : \{0, 1\}^n \rightarrow \{0, 1\} \) and any interpretation, we can define a function \( g^* : \{0, 1, *\}^n \rightarrow \{0, 1\} \) that is identical to this interpretation. In this sense, the treatment we suggest in this chapter is at least as general as the one hinted upon in Section 6.1 (and elaborated on in (Khardon and Roth, 1994c)). We show now that it is actually strictly more expressive than the interpretations discussed earlier in Section 6.1.

Consider first the \textit{universal interpretation}. In this case, given a partial assignment \( v' \in \{0, 1, *\}^n \) that satisfies the function \( g \) we interpret it as: for all possible extensions \( v' \) of \( v \) to total vectors, \( g(v') = 1 \). It is easy to define a function \( f : \{0, 1, *\}^n \rightarrow \{0, 1\} \) which is not a universal interpretation of any Boolean function. Moreover, \( f \) can be a very simple function, e.g., a conjunction. For example, consider the conjunction \((x_1 = 1) \land (x_2 = 0) \land (x_3 = *)\). The partial assignment \( v = (x_1 = 1, x_2 = 0) \) is mapped by it to 1 while the partial assignment \( v = (x_1 = 1, x_2 = 0, x_3 = 0) \) is mapped by it to 0. Clearly, no universal interpretation is consistent with this.

In the case of the \textit{existential interpretation}, given a partial assignment \( v' \in \{0, 1, *\}^n \) that satisfies the function \( g \) we interpret it as: there exists an extension \( v' \) of \( v \) to total vectors such that \( g(v') = 1 \). Consider the function \( f : \{0, 1, *\}^n \rightarrow \{0, 1\} \) which maps \( v = (v_1 = 1, v_2 = 1, v_3 = *) \) to 1 and all the other points in \( \{0, 1, *\}^n \) to 0. Clearly, no existential interpretation of a Boolean function is consistent with \( f \). The representation we have decided to use in this chapter is therefore richer than previously used representations. The main gain, however, of the richer representation, as was shown previously, is that it allows for a natural interpretation of the problem of reasoning from incomplete information, and thus capture more than just deductive tasks.

The work that is most relevant to our current study is that of Schuurmans and Greiner (Schuurmans and Greiner, 1994). This paper studies the problem of learning "default concepts", from partial assignments that are classified as either positive or negative examples. In this model, examples are drawn according to some probability distribution, and then a (probabilistic or arbitrary) "blocking process" hides some of the attributes. (This corresponds to the existential interpretation from above.) There are some major differences between our approach and the approach suggested
The main difference is that they are interested in learning “default concepts” over \(\{0, 1\}^n\) in order to augment a knowledge base, and then reason with it using an existing default reasoning formalism. We, on the other hand, do not separate between the knowledge base and the default concepts. Rather, we learn “generalized” concepts that support the type of non-monotonic inference we want to capture. Also, the knowledge representation used there, of functions over \(\{0, 1\}^n\), is different from ours.

### 6.6 Concluding Remarks

We have presented a new approach to the problem of reasoning with incomplete information within the Learning to Reason framework. In addition to our view that reasoning should be studied as an inductive phenomenon, we argued that incomplete information in the interface of the agent with her environment is not a “necessary evil”, but rather helpful: we took the view that missing information in the interaction of the agent with the environment may be as informative as observed information.

Most of this chapter was devoted to formalizing this intuition and showing that we can present the problem of reasoning with incomplete information as a problem of learning attribute functions over the domain \(\{0, 1, \ast\}^n\). We developed a theory for inference in the presence of incomplete information and have shown that reasoning with it is efficient, and that it avoids many of the representational problems which existing default reasoning formalisms struggle with.

We view the large body of research on defeasible theories of reasoning as an attempt to characterize the type of defeasible reasoning people do. While there is today some understanding of human-like patterns of reasoning, we believe that no definition can characterize the type of behavior expected, given some body of partial knowledge as a starting point. The Learning to Reason framework suggests a more “operational” definition, that is yet rigorous and amenable for theoretical study. As we have argued here, it can be shown to match our expectations in cases in which the reasoning problem is well defined. This has been illustrated here in cases of deduction and, more interestingly, in cases of “complete understanding” of the type of partial knowledge we have, as in the examples analyzed in this chapter.

As mentioned before, determining how complex the attribute functions in naturally arising reasoning problems are, and whether those can indeed be represented as simple functions over \(\{0, 1, \ast\}^n\), is an important empirical question. This and other empirical and theoretical questions that our work raises are discussed further in the concluding chapter.
We have presented in this thesis a computational study of commonsense reasoning. Our complexity-theoretic analysis of current theories of reasoning was used to argue that these are inadequate for commonsense reasoning. Motivated by this we developed a new framework for the study of reasoning, in which a learning component has a principal role. We have shown that this approach efficiently supports “more reasoning” than traditional approaches and at the same time matches our expectations of plausible patterns of reasoning in cases where other theories do not.

The approach developed is aimed at overcoming the main computational difficulties in the traditional treatment of reasoning, which stem from its separation from the “world”. Indeed, we have shown that the Learning to Reason approach developed here provides a true increase in the reasoning power of the agent, and have presented several positive results that do not hold in the traditional setting.

To support our efficient reasoning we use the notion of a model-based representation which is developed and analyzed here. In addition to its computational advantages, we show that this representation suggests an intriguing view on reasoning, and in particular, deductive reasoning. Using a model-based representation, our agent behaves logically (to an outside observer), even though her knowledge representation consists of a set of models and not a logical formula, and she does not use any logic or “theorem proving”.

The traditional way to represent knowledge and abstract the reasoning problem can be criticized not only on computational but on even the more fundamental grounds that it does not capture what people view as plausible patterns of reasoning. In the final part of this thesis, we developed a new approach to the study of the non-monotonicity of human commonsense reasoning, within the Learning to Reason approach. The theory developed was shown to support efficient reasoning with incomplete information, and to avoid many of the representational problems which existing default reasoning formalisms face.

We have shown how the various reasoning tasks considered in this thesis relate
to each other and concluded that all the reasoning tasks are supported together naturally. In order to derive a conclusion our agent employs her inference procedures, using the knowledge representations she has learned in her interactions with the world. Whether the task looks to an outside observer as pure deduction, or as inference “with defaults” is determined mainly by what the observer knows about the world.

Unlike the “static” approaches to the study of reasoning within the knowledge-based system approach, the approach developed in this thesis suggests an “operational” approach to studying reasoning. Nevertheless, the approach is rigorous and amenable to analysis. As we have argued, this approach efficiently supports a lot “more reasoning” than traditional approaches and at the same time it matches our expectations of plausible patterns of reasoning in cases where other theories do not.

This work suggests several areas in which further theoretical study is needed, as well as some interesting questions for empirical study.

Clearly, an important step would be to extend the theory to support other reasoning tasks. In particular, developing a theory for planning, or reasoning about actions seems to be an important step. The theory should also be extended to support more expressive representations. Extending it to a probabilistic domain, and beyond the pure propositional case, to allow relations, for example, are important steps in this direction.

On the learning side, algorithms that can learn efficiently in the presence of irrelevant attributes have a central role in the theories developed here (and in particular, in the discussion in the previous chapter) and more work is needed to advance our understanding of these issues. Also, this work suggests a view of unsupervised learning. In most previous studies of unsupervised learning, the goal of the learner is to output a probability distribution that is as close as possible (in a well defined sense) to the target distribution. In the framework presented here the agent also learns the “world” (or, better, the state of “truth” in the “world”) in an unsupervised manner (this is even more explicit in the final chapter) and is then able to answer queries about it. The major difference is that we are interested in a strictly easier (but sufficient, we believe) task, of performing well on a reasoning task. More work is needed in order to put this framework in a probabilistic context and understand the relations to other theories of unsupervised learning.

However, we believe that the most fundamental theoretical problem suggested by our approach is the problem of the interface. The type of interface the agent has with the environment is a crucial component of the Learning to Reason approach in general, and in particular, of the treatment of reasoning with incomplete information presented. We use this latter approach to illustrate the interface problem. In a random situation we assume that the information required for learning the attribute functions is supplied by the oracle $EX(D)$. This oracle might supply, for example, observations like $(\text{bird} = 1, \text{penguin} = 1, \text{fly} = 0)$. Our treatment of
specificity, and of conflicting defaults in general, relies therefore on the knowledge that a penguin is also a bird. In general, however, it is possible that in an observed example the attribute penguin is 1 but the attribute bird is *. That is, our model of the interface implicitly assumes some amount of knowledge. We speculate that one could model the interaction as an entity with two (or more) layers. The first layer interfaces directly with the environment. The second layer, receives examples in \{0, 1, *\}^n from the first layer and uses the current knowledge of the system, that is, the attribute-function representation, to modify the observations and transfer them to the agent. The output of this second layer is the interface we discuss in Chapter 6. We call this process a completion process. In this process, the attribute function of every *-valued attribute is evaluated, and if the evaluation is different from *, the value of this attribute is modified before it is being used in the learning process. The problem of modeling the interface with the environment calls for a better understanding of incremental learning, in which, in a process of continuous learning, knowledge accumulated in early learning stages is used in later stages.

We end on an empirical note: Throughout the thesis we have made assumptions with regard to the complexity of classes of queries and other representations. For example, we have considered many default reasoning examples used in the literature and have shown that they all have very simple attribute-function representations which are efficiently learnable. Determining how complex those representations are in naturally arising reasoning problems is an important empirical question.

Perhaps the major difference between the knowledge-based system approach to reasoning and the Learning to Reason approach is that our approach suggests that in order to make theories of reasoning work in practice, we need to train them over a large number of examples. Therefore, finding good and large test beds on which to validate this theory is one of the most important next steps.


Reiter, R. 1980. A logic for default reasoning. Artificial Intelligence, 13(1,2).


