Notes on logic and probability

with a view towards statistical relational learning

Vera Koponen
Department of Mathematics
Uppsala University

December 10, 2023

Contents

Chapter 1

Introduction

This text aims at giving a reader who is familiar with the syntax and semantics of propositional and first-order logic (also called predicate logic) an understanding of the basic principles of combining logic and probability in ways that are relevant within the field of *statistical relational learning*, also called *statistical relational artificial intelligence* or *probabilistic logic learning*. It also introduces the reader to the core ideas in statistical relational learning such as learning a probabilistic model from a set of possibly multidimensional data and using the model for predicting the probability of an event expressed by a first-order formula on a new domain of objects. Statistical relational learning, which is a subfield of artificial intelligence (AI), has been characterized as "the intersection of probabilistic reasoning, logical representations, and machine learning" [2, p. 31]. From a more general perspective, it has also been said that "AI research arguably builds on two formal foundations: probability and logic" [3, p. 17]. My hope is that, after having absorbed the material of this text, the reader will have a solid foundation for learning more about statistical relational learning. I also hope that the presentation here will help the reader to see the common ideas behind a number of different logic-based formalisms within the area of statistical relational learing. For further reading, books and survey articles about statistical relational learning that I am aware of include these: [1, 2, 3, 4, 9, 16, 17, 30].

Formal (or symbolic) logic provides formal languages (syntax and semantics), in particular propositional logic and first-order logic, for representing knowledge and reasoning about it in a precise way. This assumes that knowledge is certain. This is the case if, for example, we study a mathematical theory the axioms which are known and do not suddenly change. As another example, in the area called *formal methods* [26] one uses some logical formalism (often propositional or first-order logic) to describe some large and complex system with known and deterministic components and then the goal is to verify (using the logical formula describing the system) that the system satisfies certain conditions. (If one can not verify it, it is safer to redesign the system.) One difficulty in this area is that the problem of deciding whether an arbitrary propositional formula is decidable is NP-complete, and the problem of determining if an arbitrary first-order sentence is satisfiable is undecidable; see Chapter 2 for more about this. It is worth mentioning that first-order logic appears in other disguises, for example as the programming languages Prolog and SQL (the later is an abbreviation of *Structured Query Language* which is used for querying databases); these languages are based on the semantics of first-order logic but have a different syntax.

If information is uncertain, or if we only have partial information but we still want to make predictions that go beyond the information that we have, then methods from formal logic alone are not enough. The situation where information is uncertain or incomplete is studied within the field of machine learning. *Machine learning* can roughly be described as the study of sytems that improve their behaviour (for example predictive accuracy) as they increase their experience (i.e. as they get more information). Probability theory and statistics provide tools for dealing with uncertainty and for making statistical predictions from samples, so, unsurprisingly, probability and statistics play a central role in machine learning. However, probability and statistics do not provide the tools for systematic representation of complex knowledge or complex (possibly multidimensional) relations between objects. This motivates a quest for combining logic and probability. However, the desire to combine logic and probability predates machine learning by a few centuries. According to Hailperin [7] it goes back at least to Leibniz (1646–1716) and after that the question has been considered by, among others, De Morgan , Boole, Keynes and Carnap. I refer to Hailperin [7] for references to various texts back in time on this issue.

The main parts of this text are Chapters 4 and 5. Chapter 4 discusses logic and probability. All reasoning about logic and probability that I have seen in the field of artificial intelligence is based on two approaches to combining logic and probability. One approach is the "probabilities on possible worlds approach". In this approach every possible world has a probability and the probability of a propositional formula, or first-order sentence, is the sum of the probabilities of all possible worlds in which the formula is true. In propositional logic a possible world is a truth assignment and in first-order logic a possible world is a first-order structure. The other approach is the "probabilities on a domain approach", or "statistical information approach". In this approach we have a probability distribution on the objects of a domain (i.e. a set of objects) D which, via the product measure, induces a probability distribution on all n-tuples of elements from D (for every n) and this in turn induces a probability of any first-order formula $\varphi(x_1, \ldots, x_n)$ with free variables $x_1, \ldots, x_n$ with respect to a first-order structure with domain D.

The clearest discussion about these two approaches to logic and probability that I am aware of is the one by Halpern in [8]. I follow Halpern in this text but my definitions are less general than his. My choice of specializing his definitions to a more narrow context is based on the hope that it will make this text more accessible to a wider audience (having less experience with fomal logic and mathematics); and still my definitions are general enough to cover all uses within statistical relational learning that I have seen. Although Halpern [8] works within a context more general than that of this text, the notions of "probabilities on possible worlds", or type-2-probability-structures with Halpern's terminology, and "probabilities on a domain", or type-1-probability-structures with Halpern's terminology, have been generalized. The concept of *measure team* (and *measure team logic*) by Hyttinen, Paolini and Väänänen [12] generalizes Halpern's concept of type-1-probability-structure. The concept of *probabilities on first-order sentences* by Gaifman [6] generalizes Halpern's concept of type-2-probability-structure (because every type-2-probability-structure immediately induces a probability on first-order sentences in the sense of Gaifman).

Chapter 5 introduces the reader to the field of statistical relational learning. In statistical relational learning a typical problem is to "learn" a probabilistic model from a set of data, where the set of data consists of a set of objects (a domain) together with relations between the objects and/or properties that objects may have. Such a set of data is naturally represented by a first-order structure (i.e. a structure in the sense of first-order logic). The probabilistic models considered in statistical relational learning are often so-called graphical models. A *graphical model* is a graph (directed or undirected) the nodes of which are random variables together with additional information. The graph structure describes (conditional) (in)dependencies and the "additional information" describes (conditional) probabilities that are sufficient for determining the joint probability distribution on the set of all random variables in the graphical model. In this text I will only consider the graphical model called *Bayesian network*, for at least three reasons: the first one is that Bayesian networks is one of the best known graphical models, the second is that I want to keep this text relatively brief, and the third one is that I believe that it is easier for the reader to follow the main ideas if I only use Bayesian networks as the graphical model of choice. When constructing a Bayesian network (as in Section 5.1) we use the "probabilities on domains approach" discussed in Chapter 4. When having a probabilistic model, a Bayesian network, the next task is of course to use it to predict probabilities of events, in our context any event that can be described by a first-order formula, on other domains that the domain used for learning the probabilistic model. The "probabilities on possible worlds approach" gives a firm mathematical ground for making such predictions. By means of examples it is illustrated in Chapters 4 and 5 how to use a Bayesian network to predict probabilities on any finite domain. It is possible to give a general description/definition of the procedure, but since it is rather complicated I have chosen to illustrate the idea by examples and I hope that the reader can see how to use it in other situations than the examples in this text.

Besides the topics of Chapters 4 and 5, described above, this texts touches upon three other topics. Chapter 2 gives a brief overview of basic results about the existence or nonexistence of (efficient) algorithms which decide (some variants of) the satisfiability problem for propositional formulas, or for first-order sentences. Chapter 3 is a brief introduction to the basic ideas in the field of *inductive logic programmig (ILP)*. In ILP a typical problem is, given a set of "positive" examples and, sometimes, a set of "negative" examples", to find a *logic program* (a formula in conjunctive normal form) which corresponds to these sets of examples. The sought after logic program can be seen as a set of rules that are satisfied by all "positive" examples and not satisfied by any "negative" example; moreover, given a set of "facts" one can use the logic program to derive (via its rules) consequences of the given facts. ILP in itself does not involve probabilistic reasoning, but it is natural to combine ILP with probabilistic methods and reasoning and ILP is therefore considered as a part of the toolkit of statistical relational learning. See for example [3, 16].

Chapter 6 considers the problem of *scalability* which is of practical importance. Roughly speaking a method of inference is scalable if the computational cost (measured by time or memory space) of using it does not increase very much if the domain on which we make inferences increases. This is an important issue if one wants to make inferences on large domains such as large human populations or large sets of data. In Chapter 6 one approach to tackling the problem of scalability is briefly exposed. The approach relies on concepts and methods from *finite model theory* (a branch of logic) such as *convergence results* for events definable by logical formulas and *"almost sure elimination of quantifiers"*.

1.1 Notation and terminology

1.1.1 General

We let $\mathbb{N}$ and $\mathbb{N}^+$ denote the set of nonnegative integers and the set of positive integers, respectively. If A and B are sets then $A \cup B$ ad $A \cap B$ union and intersection of A and B, respectively. By $A \setminus B$ we denote the set $\{a \in A : a \notin B\}$, in other words the set containing all elements which belong to A but not to B. By $|A|$ we denote the *cardinality* of the set A. If A is finite (as is usually the case in this text) then $|A|$ is the number of elements in A which we may also just call the "size" of A. If $\vec{a}$ is a finite sequence/tuple of objects (of whatever kind), then $|\vec{a}|$ denotes the length of the sequence/tuple. Sometimes I use the symbol ':=' to emphasize that the notation to the left of ':=' is defined to denote whatever is written to the right of ':='.

1.1.2 Propositional logic

Familiarity of the syntax and semantics of propositional logic and of first-order logic (also called predicate logic) is assumed. If $\mathbf{P}$ denotes a set of propositional variables (atomic propositional formulas), then $\mathsf{Prop}(\mathbf{P})$ denotes the set of all propositioonal formulas that can be constructed from the propositional variables in $\mathbf{P}$ by using the connectives $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$ (and parantheses when necessary). If we like we can consider $\bot$ and $\top$ as particular atomic formulas where $\bot$ is always false (i.e. false under every truth assignment) and $\top$ is always true. Sometimes we abbreviate $\mathsf{Prop}(\{p_1, \ldots, p_m\})$ by $\mathsf{Prop}(p_1, \ldots, p_m)$. Truth assignments to a sequence $p_1, \ldots, p_m$ of propositional variables are represented by binary vectors/tuples/sequences $\vec{a} = (a_1, \ldots, a_m) \in \{0, 1\}^m$. If $\vec{a} = (a_1, \ldots, a_m)$ is a binary vector then, for every $i = 1, \ldots, m$, $a_i = 1$ means that $\vec{a}$ assigns p_i the value "true" and $a_i = 0$ means that $\vec{a}$ assigns p_i the value "false". For $\varphi \in \mathsf{Prop}(\mathbf{P})$ and $\vec{a} \in \{0, 1\}$, the notation $\vec{a} \models \varphi$ means that φ is satisfied (or made true) by the truth assignment $\vec{a}$. The terminology *$\vec{a}$ is a model of φ* means the same as $\vec{a} \models \varphi$. We recall some more terminology regarding propositional logic.

Definition 1.1.1 All formulas in this definition are assumed to belong to $\mathsf{Prop}(\mathbf{P})$ for some set $\mathbf{P}$ of propositional variables.

(a) A formula is called *atomic* (or an *atom*) if it is a propositional variable (i.e. belongs to $\mathbf{P}$).

(b) A formula is called a *literal* if it is atomic or a negation of an atomic formula.

(c) A disjunction of literals, that is, a formula of the form $l_1 \vee \ldots \vee l_n$ where all l_i are literals, is called a *clause*.

(d) A formula with the form $\psi_1 \wedge \ldots \wedge \psi_n$ where each ψ_i is a clause is called a *conjunctive normal form (CNF)*.

(e) A formula with the form $\psi_n \vee \ldots \vee \psi_n$ where each ψ_i is a conjunction of literals (that is, has the form $\theta_1 \wedge \ldots \wedge \theta_m$ where each θ_j is a literal) is called *disjunctive normal form (DNF)*.

(f) A disjunction of literals (i.e. a clause) $l_1 \vee \ldots \vee l_n$ is called a *Horn clause* if at most one of $l_1, \ldots, l_n$ is atomic.

(g) A conjunction of Horn clauses, that is, a CNF $\psi_1 \wedge \ldots \wedge \psi_n$ where each ψ_i is a Horn clause, is called a *Horn formula.*

If φ is a propositional formula and T is a set of propositional formulas, then $T \models \varphi$ means that φ is a logical consequence of T, which in turn means that every truth assignment which satisfies all formulas in T also satisfies φ. Recall that $\psi_1, \ldots, \psi_n \models \varphi$ if and only if the formula $(\psi_1 \wedge \ldots \wedge \psi_n) \to \varphi$ is a tautology (i.e. true under every truth assignment).

1.1.3 First-order logic

By a first-order language, often denoted L, we mean a set of constant symbols, relation symbols and/or function symbols. An *L-formula* is a first-order formula that is constructed from symbols in L (and of course the logical symbols $\neg, \wedge, \vee, \rightarrow, \leftrightarrow, \forall, \exists$, (object) variables, often denoted $x, y, z, \ldots$, and auxilliary symbols such as the comma symbol and parantheses). We sometimes denote the set of all L-formulas by $FO(L)$ ('FO' for 'first-order'). A formula is called *quantifier-free* if it contains no quantifier (i.e. if it has no occurence of $\exists$ or $\forall$). First-order structures will be denoted by calligraphic letters $\mathcal{A}, \mathcal{B}, \ldots, \mathcal{M}, \mathcal{N}$ and their domains are often denoted by the corresponding noncalligraphic letter $A, B, \ldots, M, N$. An *L-structure* $\mathcal{A}$ is a first-order structure such that all symbols in L have interpretations on the domain of $\mathcal{A}$.

A (first-order) *sentence* is a (first-order) formula without free variables; such a formula is also called a *closed formula.* In computer science the word *grounded formula* means the same as sentence. The terminology "grounded formula" is particularly common (in computer science) when speaking of sentences without quantifiers ($\forall$, $\exists$). A first-order formula is called *quantifier-free* if it contains no quantifiers. If a first-order formula is denoted by $\varphi(x_1, \ldots, x_n)$ then it means that all free variables of the formula belongs to the sequence $x_1, \ldots, x_n$, but we *do not* insist that every x_i occurs in φ; so when writing $\varphi(x_1, \ldots, x_n)$ some of the variables displayed in $x_1, \ldots, x_n$ may be redundant, also called "dummy variables". When denoting a formula by $\varphi(x_1, \ldots, x_n)$ then we assume, unless something else is said, that all variables in the sequence $x_1, \ldots, x_n$ are different (similarly if we denote a formula by $\varphi(\vec{x})$ then we assume that all variables in the sequence $\vec{x}$ are different).

If φ is an L-sentence and $\mathcal{A}$ is an L-structure, then the notation $\mathcal{A} \models \varphi$ means that φ is true (or satisfied) in the structure $\mathcal{A}$; in this case we also say that $\mathcal{A}$ is a *model of* φ. The word "model" has many different meanings. A "probabilistic model", such as a Bayesian network, is not the same as a first-order model of some first-order sentence. Hopefully it will be clear from the context what I mean if I use the word "model". An L-sentence is called *valid* if it is true in every L-structure. If φ is an L-sentence and T is a set of L-sentences, then $T \models \varphi$ means that every L-structure which satisfies every sentence in T also satisfies φ (or differently formulated, every model of T is a model of φ. Analogously to propositional logic we have that if $\psi_1, \ldots, \psi_n, \varphi$ are L-sentences, then $\psi_1, \ldots, \psi_n \models \varphi$ if and only if $(\psi_1 \wedge \ldots \wedge \psi_n) \to \varphi$ is valid.

Let $L' \subseteq L$ be first-order languages, $\mathcal{A}'$ an L'-structure and $\mathcal{A}$ an L-structure. If for every symbol $S \in L'$ the interpretation of S in $\mathcal{A}'$ (denoted $S^{\mathcal{A}'}$) coincides with the interpretation of S in $\mathcal{A}$ (that is, $S^{\mathcal{A}'} = S^{\mathcal{A}}$), then one calls $\mathcal{A}$ and *expansion* of $\mathcal{A}'$ (to L) and one calls $\mathcal{A}'$ a *reduct* of $\mathcal{A}$ (to L'). Note that for every L-structure $\mathcal{A}$ there is (still assuming $L' \subseteq L$) a unique L'-structure $\mathcal{A}'$ such that $\mathcal{A}'$ is the reduct of $\mathcal{A}$ to L'. This structure $\mathcal{A}'$ is denoted $\mathcal{A} \upharpoonright L'$.

Let $\varphi(x_1, \ldots, x_n)$ be an L-formula and let $a_1, \ldots, a_n \in A$ where $\mathcal{A}$ is an L-structure. The notation $\mathcal{A} \models \varphi(a_1, \ldots, a_n)$ means that if we enlarge the language L to $L^* = L \cup$

$\{\hat{a}_1, \ldots, \hat{a}_n\}$, where $\hat{a}_1, \ldots, \hat{a}_n$ are new constant symbols, and if $\mathcal{A}^*$ is the expansion of $\mathcal{A}$ to the language L^* where $\hat{a}_i$ is interpreted as a_i for all $i = 1, \ldots, n$, then $\mathcal{A}^* \models \varphi(\hat{a}_1, \ldots, \hat{a}_n)$. We also use the following notation:

$$\varphi(\mathcal{A}) := \{(a_1, \ldots, a_n) \in A^n : \mathcal{A} \models \varphi(a_1, \ldots, a_n)\}.$$

Definition 1.1.2 Let L be a first-order language.

(a) An L-formula is called *atomic* (or an *atom*) if it it has the form '$s = t$' where s and t are L-terms (for example variables or constant symbols) or the form '$R(t_1, \ldots, t_k)$' where R is a relation symbol of arity k and $t_1, \ldots, t_k$ are L-terms.

(b) The notions of *literal, clause, conjunctive normal form (CNF), disjunctive normal form (DNF), Horn clause* and *Horn formula* are defined as the corresponding notions for propositional logic (Definition 1.1.1), but using the definition of atomic formula in the first-order case (the previous item (a)).

Chapter 2

Logic and decidability

The meaning of the word “algorithm” is roughly a “finite set of instructions”, like for example the the algorithm for computing the greatest common divisor of two integers. However, the expression “finite set of instructions” is not a mathematically precise concept. There are several mathematical formalizations of the notion of algorithm (i.e. precise ways of describing algorithms). The most well-known formalization is probably the *Turing machine* concept. (There are many books which define Turing machines, for example [24, 28, 31].) All theorems in this section become precise mathematical statements if we interpret “algorithm” as meaning “Turing machine”.

If we want to be quite formal we could define a *decision problem* to be a function $f : W \to \{0, 1\}$ where $W \subseteq \Sigma^*$ and Σ^* denotes the set of all strings of symbols from an (usually finite) alphabet Σ. For example, if $\Sigma = \{0, 1\}$ then 101 and 0011001 are strings in Σ^*. Every propositional formula as well as every first-order formula is a string of symbols. Although we may have an infinite supply of propositional variables $p_1, p_2, p_3, \ldots$ (which are seen as just symbols) or of first-order variables $x_1, x_2, x_3, \ldots$ we can code them with a finite alphabet, for example $\{0, 1\}$. As is well-known, in real computers all expressions that we can type with the keyboard are coded as binary strings in the computer. Thus every first-order formula that we can describe, with for example the typesetting languge LaTeX, can be coded as a binary string.

An example of a decision problem in logic is this. Let $\mathbf{P} = \{p_1, p_2, p_3, \ldots\}$ be a set of propositional variables and recall that $\mathsf{Prop}(\mathbf{P})$ denotes the set of all propositional formulas that can be constructed with the propositional variables in $\mathbf{P}$, the connectives $\neg, \wedge, \vee, \to, \leftrightarrow$ and parantheses ‘(’ and ‘)’. Now the problem is to decide, for any $\varphi \in \mathsf{Prop}(\mathbf{P})$, if φ is satisfiable. We can describe this decision problem formally by letting $\Sigma = \mathbf{P} \cup \{\neg, \wedge, \vee, \to, \leftrightarrow, (,)\}$ and letting W be the set of all strings in Σ^* that are propositional formulas. Then, for every $\varphi \in W$, we let $f(\varphi) = 1$ if φ is satisfiable and $f(\varphi) = 0$ otherwise. The alphabet Σ as we defined it is infinite which makes it impossible for a Turing machine to work with, since a Turing machine can only work with a finite alphabet. But as mentioned above, we can replace Σ by a finite alphabet, say $\{0, 1\}$, and then we can replace W with the set of all binary codes of propositional formulas. The function f is defined in the same way if we interpret φ as the binary code of a propositional formula.

Definition 2.0.1 Informally, we say that a decision problem is *decidable* if there is an algorithm which solves it. More formally, a decision problem $f : \Sigma^* \to \{0, 1\}$ is *decidable* if there is an algorithm which computes f. And most formally, a decision problem $f : \Sigma^* \to \{0, 1\}$ is *decidable* if there is a Turing machine which computes f. A decision problem which is not decidable is called *undecidable*.

Definition 2.0.2 Let A denote an algorithm which takes strings of symbols over some alphabet Σ as input and which terminates on every input with output 0 or 1. We say that A is *efficient* if there is a polynomial $p(x)$ such that, for every input string w, A terminates after at most $p(|w|)$ steps (where $|w|$ denotes the length of w).

2.1 Propositional logic, decidability and efficiency

Theorem 2.1.1 *Let* $\mathbf{P}$ *be a (possibly infinite) set of propositional variables. The following problems are decidable:*

(a) Given arbitrary $\varphi \in \mathsf{Prop}(\mathbf{P})$ *is* φ *satisfiable?*

(b) Given arbitrary $\varphi \in \mathsf{Prop}(\mathbf{P})$ *is* φ *a tautology?*

(c) Given arbitrary $\varphi_1, \ldots, \varphi_n, \psi \in \mathsf{Prop}(\mathbf{P})$ *is* ψ *a logical consequence of the formulas* $\varphi_1, \ldots, \varphi_n$*?*

The perhaps simplest way, from a conceptual point of view, of deciding the problems in Theorem 2.1.1 is to use truth tables. However this method may take exponential time (compared with the length of the formula) in the worst case, because if φ contains k different propositional variables, then the whole truth table for φ will have 2^k lines. For each one of problems (a)–(c) in Theorem 2.1.1, *no* efficient algorithm is known. All of the problems (a)–(c) of the same theorem are closely related[1] and are known to be *NP-complete* (where I refer to [28, 31], for example, for a definition of 'NP-complete').

Let φ is a DNF and $\vec{a}$ a truth assignment, then $\vec{a}$ satisfies φ if and only if $\vec{a}$ satsifies (at least) one of the conjuncts of φ; thus to decide satisfiability one just checks the conjuncts for satisfiability, one by one. Since the number of conjuncts is less than the length of φ it follows that one can decide whether an arbitrary DNF φ is satisfiable in linear time in the length of φ (so this algorithm is efficient). However, in Section 3 we will see that CNF's play an important role in logic programming and artificial intelligence. No efficient algorithm is known that decides whether an arbitrary CNF is satisfiable; in fact, this is an NP-complete problem. However, for the subclass of CNF's called Horn formulas (see Definition 1.1.1) we have the following positive result (see for example [28, Theorem 4.2]) which is relevant for logic programs.

Theorem 2.1.2 *There is an efficient algorithm which decides the following problem: Given an arbitrary Horn formula* φ*, is it satisfiable?*

Remark 2.1.3 Although no efficient algorithm has been found (and many doubt it exists) for the satisfiability problem in general (or for CNF's) it turns out that formulas that occur in practical applications can often be checked for satisfiability in polynomial time. Much research has been done around the problem of finding algorithms that are efficient for various subclasses of propositional formulas.

2.2 First-order logic and decidability

Theorem 2.2.1 *The following problem is decidable: Given an first-order sentence without quantifiers, does it have a model?*

[1] For example, a formula φ is a tautology if and only if $\neg\varphi$ is not satisfiable.

If we do not use function symbols then the above theorem can be generalized a bit.

Theorem 2.2.2 *The following problem is decidable: Given a sentence φ of the form*

$$\exists x_1, \dots, \exists x_n \forall y_1, \dots, \forall y_m \theta(x_1, \dots, x_n, y_1, \dots, y_m)$$

where θ is quantifier-free and no function symbol occurs in θ, does φ have a model?

Proof sketch. Suppose that $\mathcal{M} \models \varphi$ (where the domain of $\mathcal{M}$ may be infinite). We may assume that $\mathcal{M}$ is an L-structure where L is the language that contains all relation symbols and contant symbols that occur in φ, but no other relation or constant symbols. Then there are $a_1, \dots, a_n \in M$ (where M is the domain of $\mathcal{M}$) such that

$$\mathcal{M} \models \forall y_1, \dots, \forall y_m \theta(a_1, \dots, a_n, y_1, \dots, y_m).$$

Let $\mathcal{N}$ be the substructure of $\mathcal{M}$ with domain

$$N = \{a_1, \dots, a_n, c_1^{\mathcal{M}}, \dots, c_k^{\mathcal{M}}\}$$

where $c_1, \dots, c_k$ are all constant symbols that occur in θ. (If φ had function symbols we could not be sure that $\mathcal{M}$ had a substructure with finite domain, let alone with domain N.) From the choice of $a_1, \dots, a_n$ it follows that

$$\mathcal{N} \models \forall y_1, \dots, \forall y_m \theta(a_1, \dots, a_n, y_1, \dots, y_m)$$

and hence $\mathcal{N} \models \varphi$. Note that $|N| \leq n+$ "the number of constant symbols in φ", so $|N| \leq |\varphi|$ (where $|\varphi|$ denotes the length of φ).

We have shown that if φ has a model then it has a finite model with domain of size at most $|\varphi|$. So to check if φ has a model it suffices to check if it has a model with domain of size at most $|\varphi|$. Since there is, up to isomorphism (which preserves satisfiability), only finitely many L-structures with domain of size at most $|\varphi|$, an algorithm can check if φ has a model by simply checking satisfiability in finitely many L-structures. (Unfortunately, this algorithm is inefficient.) $\square$

Remark 2.2.3 (a) A model/structure is called *finite* if its domain is finite. If we change "does it have a model" to "does it have a finite model" in the problems in Theorems 2.2.1 and 2.2.2, then the problems remain decidable (with essentially the same proofs).
(b) Observe that Theorem 2.2.2 covers the special cases when φ has the form $\forall x_1, \dots, \forall x_n \theta(x_1, \dots, x_n)$ or the form $\exists y_1, \dots, \exists y_m \theta(y_1, \dots, y_m)$ with θ without any quantifier or function symbol.

If we use only relation symbols but not the identity symbol '$=$' then have decidability for formulas with a prefix of quantifiers of the form '$\forall x_1, \forall x_2 \exists y_1, \dots, \exists y_m$' followed by a quantifier-free formula, as stated by the following result:

Theorem 2.2.4 *([5, Corollary 5.2.3]) The following problem is decidable: Given a first-order sentence φ of the form*

$$\forall x_1, \forall x_2 \exists y_1, \dots, \exists y_m \theta(x_1, x_2, y_1, \dots, y_m)$$

where θ has no quantifier, no constant symbol, no function symbol and does not use the symbol '$=$', does φ have a model?

From the proof of the above theorem (see [5, Corollary 5.2.3]) it follows that if we replace "does φ have a model" by "does φ have a finite model' in Theorem 2.2.4 then the statement remain true. For more about decidability of fragments of first-order logic and related "description logics" used in applications, see for example [11].

If we consider first-order sentences in general then we loose decidability.

Theorem 2.2.5 *The following two problems are undecidable:*

(a) Given a first-order sentence, does it have a model?

(b) Given a first-order sentence, does it have a finite model (i.e. one with finite domain)?

Remark 2.2.6 (a) Since φ is valid (i.e. true in all L-structures where L is the langauge of φ) if and only if $\neg\varphi$ does not have a model, it follows that (a) in Theorem 2.2.5 remains undecidable if "does it have a model" is replaced by "is it valid". By Gödel's completeness theorem, we can also replace "does it have a model" in (a) by "is there a formal proof of it" and the problem remains undecidable. That the problem to determine if a first-order sentence has a formal proof is undecidable is proved in [19, Chapter XIV, Theorem 54]. This implies the other mentioned variants of the undecidability of (a).

(b) That (b) in Theorem 2.2.5 is undecidable is refered to as Trahtenbrot's theorem which is proved in (for example) [5, Theorem 7.2.1]. The problem (b) remains undecidable if "does it have a finite model" is replaced by "is it true in all finite structures (for the same language as φ)".

Remark 2.2.7 By Theorem 2.2.5 the satisfiability problem is undecidable for first-order sentences in general. However, similarly to the situation for propositional logic, discussed in Remark 2.1.3, first-order sentences arising in modeling with first-order logic in contexts outside of pure mathematics or pure logic are often "simple enough" that questions concerning satisfiability (or related issues) can be decided within reasonable time by existing algorithms. Thus, first-order logic is not doomed to uselessness with respect to algorithmic issues in spite of negative general results.

Chapter 3

Logical formulas as programs, which can be learned

This chapter is a very brief introduction to the basic ideas within the field of *inductive logic programming* (abbreviated ILP). I will only use notation from propositional or first-order logic when describing clauses or sets (or conjunctions) of clauses. Therefore my notation is different from the Prolog style syntax often used by computer scientists to describe clauses. For example, the clause $p_1 \vee \neg p_2 \vee \neg p_3$ can be written as $p_1 \text{:-}\ p_2, p_3$ with Prolog syntax. The interpretation of the later expression is that if both p_2 and p_3 are true then p_1 is true.

3.1 Logical formulas as programs for deducing consequences

The ideas of this section apply to both propositional logic and first-order logic, but when reading it for the first time it may be simpler to imagine that all formulas that are mentioned are propositional formulas (i.e. members of $\mathsf{Prop}(\mathbf{P})$ for a set $\mathbf{P}$ of propositional variables). Recall the terminology from Definition 1.1.1. Let $\gamma = l_1 \vee \ldots \vee l_n$ be a clause where each l_i is a literal. Then γ is logically equivalent to $(\neg l_1 \wedge \ldots \neg l_{n-1}) \rightarrow l_n$. But γ is also logically equivalent to $(\neg l_1 \wedge \ldots \wedge \neg l_{n-2} \wedge \neg l_n) \rightarrow l_{n-1}$. In general, for every $i = 1, \ldots, n$, γ is logically equivalent to

$$(\neg l_1 \wedge \ldots \wedge \neg l_{i-1} \wedge \neg l_{i+1} \wedge \ldots \wedge \neg l_n) \rightarrow l_i.$$

For every i, $\neg l_i$ is logically equivalent to a literal, since for any formula φ, $\neg\neg\varphi$ is logically equivalent to φ. As an example, if p, q and r are atomic formulas (propositional variables), then all the following formulas are logically equivalent:

$$p \vee \neg q \vee r,\ \ (\neg p \wedge q) \rightarrow r,\ \ (\neg p \wedge \neg r) \rightarrow \neg q,\ \ (q \wedge \neg r) \rightarrow p.$$

A formula of the form $(l_1 \wedge \ldots \wedge l_{n-1}) \rightarrow l_n$ where each l_i is a literal is also often called a *clause* in the literature.[1] In this case the literal l_n is called the *head* of the clause. Let the symbol $\perp$ denote a formula which is always false (so it can be seen as an abbreviation of $p \wedge \neg p$ for any formula p). Then we can write a clause like

$$(l_1 \wedge \ldots \wedge l_n) \rightarrow \perp$$

which is logically equivalent to $\neg(l_1 \wedge \ldots \wedge l_n)$, so both formulas express the *constraint* that it is not possible for all $l_1, \ldots, l_n$ to be true at the same time.

[1]Thus a *clause* can refer to a formula of the form $l_1 \vee \ldots \vee l_n$ or of the form $(l_1 \wedge \ldots \wedge l_{n-1}) \rightarrow l_n$ where all l_i are literals.

A clause like $(l_1 \wedge \ldots \wedge l_{n-1}) \to l_n$ expresses that if all $l_1, \ldots, l_{n-1}$ are true then so is l_n. Thus if we assume that $(l_1 \wedge \ldots \wedge l_{n-1}) \to l_n$ is true (in some context) and we also know (or assume) that the facts $l_1, \ldots, l_{n-1}$ hold, then we can draw the conclusion that l_n holds. This is just an instance of the rule called *modus ponens*, or *$\to$-elimination*, which can be formulated symbolically as

$$\frac{\varphi \qquad \varphi \to \psi}{\psi}$$

where the premisses are above the line and the conclusion below it.

When working with clauses in the form $l_1 \vee \ldots \vee l_n$ we can formulate a rule which looks a bit different but serves the same purpose. Since $l_1 \vee \ldots \vee l_n$ expresses that at leat one of l_i, $i = 1, \ldots, n$ is true, it follows that if we know that $l_1 \vee \ldots \vee l_n$ holds and we also know (or assume) that all of $l_1, \ldots, l_k$ are false, where $k < n$, then we can draw the conclusion that at least one of $l_{k+1}, \ldots, l_n$ is true, in other words that $l_{k+1} \vee \ldots \vee l_n$ holds. Symbolically we can formulate this rule as

$$\frac{\neg l_1 \wedge \ldots \wedge \neg l_k \qquad\qquad l_1 \vee \ldots \vee l_n}{l_{k+1} \vee \ldots \vee l_n}. \tag{3.1.1}$$

Or if we prefer to use '$\to$' we can translate $l_1 \vee \ldots \vee l_n$ to the logically equivalent formula $(\neg l_1 \wedge \ldots \wedge \neg l_k) \to (l_{k+1} \vee \ldots \vee l_n)$ and then use modus ponens:

$$\frac{\neg l_1 \wedge \ldots \wedge \neg l_k \qquad\qquad (\neg l_1 \wedge \ldots \wedge \neg l_k) \to (l_{k+1} \vee \ldots \vee l_n)}{l_{k+1} \vee \ldots \vee l_n}.$$

A set, or conjunction, of clauses can be seen as a "program", which we call a *logic program*, for deducing consequences if given some "facts" as input. Thus every CNF can be viewed as a logic program. In what follows I will consider clauses in the disjunctive form $l_1 \vee \ldots \vee l_n$ and I will use the rule (3.1.1). As an example let

$$\varphi \;=\; (\neg p \vee \neg r) \wedge (\neg s \vee t) \wedge (r \vee \neg t \vee q).$$

Then φ is a CNF and also a logic program. With set notation the same logic program would be denoted $\{(\neg p \vee \neg r),\ (\neg s \vee t),\ (r \vee \neg t \vee q)\}$. Let the logic program φ be given p and s as input. We only use the rule (3.1.1) to derive new conclusions. From p and the clause $\neg p \vee \neg r$ we get $\neg r$. From s and the clause $\neg s \vee t$ we get t. From $\neg r$, t and the clause $r \vee \neg t \vee q$ we get q. Thus the logic program φ has concluded that if p and s hold, then t and q hold but not r.

We do not necessarily get information about the truth or falsity of all atomic formulas. For example, if the logic program φ gets $\neg q$ and t as input, then it deduces from the clause $r \vee \neg t \vee q$ that $r \vee \neg t$ holds and in the next step, using the input t and the deduced clause $r \vee \neg t$ it can conclude that r holds. Then it can use r and the clause $\neg p \vee \neg r$ to conclude that p is false. But we do not get information about the truth value of s. In fact it is consistent with φ and the input $\neg q \wedge t$ that s is true and it is also consistent with φ and $\neg q \wedge t$ that s is false. In other words, both formulas $\varphi \wedge (\neg q \wedge t) \wedge s$ and $\varphi \wedge (\neg q \wedge t) \wedge \neg s$ are satisfiable (which the reader may verify).

It is possible that some inputs to a logic program are inconsistent with the program. Intuitively it means that that such inputs contradict the "description of the world" that the program has. For example, if φ is still the same logic program as above and we give it $\neg q$, $\neg r$ and s as input, then φ deduces (with the clause $r \vee \neg t \vee q$) that $\neg t$ and then it deduces $\neg s$ with the clause $\neg s \vee t$. But $\neg s$ contradicts the input s. Thus the input $\neg q$, $\neg r$ and s is not consistent with φ, or in other words, the formula $\varphi \wedge \neg q \wedge \neg r \wedge s$ is not satisfiable.

3.2 Learning a logic program from a set of examples, the propositional case

Let $\mathbf{P} = \{p_1, \ldots, p_m\}$ be a set of propositional variables. Truth assignments for $\mathbf{P}$ are represented by ordered m-tuples (also called sequences or vectors of length m) $\vec{a} \in \{0,1\}^m$ of zeros and ones. For example, if $\vec{a} = (1, 0, 1, \ldots)$ then $\vec{a}$ assigns p_1 the value 'true', p_2 the value 'false' and p_3 the value 'true', and so on. If $\varphi \in \mathsf{Prop}(\mathbf{P})$, that is, if φ is a propositional formula built from the propositional variables in $\mathbf{P}$, and $\vec{a} \in \{0,1\}^m$, then the notation $\vec{a} \models \varphi$ means that the truth assignment $\vec{a}$ makes φ true; in this case we can also say that *$\vec{a}$ is a model of φ.*

A subset $\mathbf{E} \subseteq \{0,1\}^m$ can be seen as a set of "examples" or "observations". A formula φ such that $\vec{a} \models \varphi$ for every $\vec{a} \in \mathbf{E}$ and $\vec{a} \models \neg\varphi$ for every $\vec{a} \in \{0,1\}^m \setminus \mathbf{E}$ can be viewed as a "complete description" of $\mathbf{E}$. We state this as a definition:

Definition 3.2.1 Let $\mathbf{E} \subseteq \{0,1\}^m$. A formula $\varphi \in \mathsf{Prop}(\mathbf{P})$ is called a *complete description*[2] *of* $\mathbf{E}$ if

- $\vec{a} \models \varphi$ for every $\vec{a} \in \mathbf{E}$ and
- $\vec{a} \models \neg\varphi$ for every $\vec{a} \in \{0,1\}^m \setminus \mathbf{E}$.

If, moreover, φ is a CNF, then, as we have seen in Section 3.1, φ is also a logic program which can be used to deduce consequences from a sequence of literals given as input. The input literals can be viewed as observations or facts from which we want to draw conclusions. Observe that φ is a complete description of $\mathbf{E}$ if and only if the following condition holds: for every truth assignment $\vec{a}$, $\vec{a} \models \varphi$ if and only if $\vec{a} \in \mathbf{E}$.

Example 3.2.2 Suppose that $m = 3$ and let us rename the propositional variables by $F = p_1$, $M = p_2$ and $H = p_3$ where we can think of F, M and H as meaning female, male and human, respectively. Suppose that some extraterrestrial has arrived to earth and made the following set observations:

$$\mathbf{E} = \{(0,1,1), (1,0,1), (0,0,0)\}.$$

Here $(0,1,1)$ represents the observation of an individual such that that individual is not female, it is male and it is human; similarly for the other binary vectors. It is easy to find a DNF which is a complete description of $\mathbf{E}$. We just take the disjunction of formulas which "express" the truth assignments in $\mathbf{E}$. The truth assignment $(0,1,1)$ is expressed by the formula $\neg F \wedge M \wedge H$, $(1,0,1)$ is expressed by $F \wedge \neg M \wedge H$ and $(0,0,0)$ is expressed by $\neg F \wedge \neg M \wedge \neg H$. The disjunction of these formulas is the following DNF:

$$\varphi = (\neg F \wedge M \wedge H) \;\vee\; (F \wedge \neg M \wedge H) \;\vee\; (\neg F \wedge \neg M \wedge \neg H).$$

Now φ is a complete description of $\mathbf{E}$, because each "block" of conjunctions is satisfied by a truth assignment in $\mathbf{E}$ and every truth assignment in $\mathbf{E}$ satisfies some block of conjuctions in φ.

However, φ is not a CNF so we cannot use it as a logic program in the way described in Section 3.1. On the other hand the following formula is a CNF and a complete description of $\mathbf{E}$:

$$\psi = (\neg M \vee H) \;\wedge\; (\neg F \vee H) \;\wedge\; (\neg F \vee \neg M) \;\wedge\; (\neg H \vee F \vee M).$$

[2] I have not seen the terminology *complete description* being used elsewhere but I think it is a convenient terminology to have.

That ψ is a complete description of $\mathbf{E}$ can be seen by checking that ψ is satisfied by exactly those truth assignments which belong to $\mathbf{E}$. There are also other ways to find a (possibly different) CNF which is a complete description of $\mathbf{E}$.

Now the extraterrestrial can use the logic program ψ as an automated procedure for drawing conclusions. For example, given F as input (a female) the extraterrestrial would use the clause $\neg F \vee \neg M$ to conclude $\neg M$ (the individual is not a male); with the same input F the extraterrestrial could also use the clause $\neg F \vee H$ to conclude H (the individual is a human). One can have objections to these conclusions. For example, some females (for example cows) are not humans. But the extraterrestrial has a very limited set $\mathbf{E}$ of observations, so one cannot expect that conclusions based only on these observations are going to be accurate in general.

Proposition 3.2.3 *Let $\mathbf{P} = \{p_1, \ldots, p_m\}$ be a set of propositional variables and let $\mathbf{E} \subseteq \{0,1\}^m$. Then there is a DNF which is a complete description of $\mathbf{E}$ and a CNF (hence a logic program) which is a complete description of $\mathbf{E}$.*

Proof. Let $\mathbf{E} \subseteq \{0,1\}^m$. For every $\vec{a} = (a_1, \ldots, a_m) \in \mathbf{E}$ let $\theta_{\vec{a}} = l_1 \wedge \ldots \wedge l_m$ where, for each $i = 1, \ldots, m$,

$$l_i = \begin{cases} p_i & \text{if } a_i = 1, \\ \neg p_i & \text{if } a_i = 0. \end{cases}$$

Now let $\varphi = \bigvee_{\vec{a} \in \mathbf{E}} \theta_{\vec{a}}$. Then φ is a DNF and a complete description of $\mathbf{E}$. To see why φ is a complete description of $\mathbf{E}$, observe, on the one hand that, for every $\vec{b} \in \mathbf{E}$, $\vec{b} \models \theta_{\vec{b}}$ so $\vec{b} \models \bigvee_{\vec{a} \in \mathbf{E}} \theta_{\vec{a}}$, and on the other hand, if $\vec{b} \models \bigvee_{\vec{a} \in \mathbf{E}} \theta_{\vec{a}}$ then there is $\vec{a} \in \mathbf{E}$ such that $\vec{a} \models \theta_{\vec{b}}$ and this is only possible if $\vec{a} = \vec{b}$ from which it follows that $\vec{b} \in \mathbf{E}$.

We can of course apply the same argument as above to the complement $\mathbf{E}^c = \{0,1\}^m \backslash \mathbf{E}$ of $\mathbf{E}$. Thus if we let $\varphi^c = \bigvee_{\vec{a} \in \mathbf{E}^c} \theta_{\vec{a}}$, then φ^c is a complete description of $\mathbf{E}^c$. Since, for every $\vec{a} \in \{0,1\}^m$, we have $\vec{a} \in \mathbf{E}$ if and only if $\vec{a} \notin \mathbf{E}^c$ it follows that $\vec{a} \models \varphi$ if and only if $\vec{a} \models \neg\varphi^c$, so $\neg\varphi^c$ is a complete description of $\mathbf{E}$. But $\neg\varphi^c$ is $\neg \bigvee_{\vec{a} \in \mathbf{E}^c} \theta_{\vec{a}}$ which is logically equivalent to $\bigwedge_{\vec{a} \in \mathbf{E}^c} \neg\theta_{\vec{a}}$. For every $\vec{a} = (a_1, \ldots, a_m) \in \{0,1\}^m$ let $\gamma_{\vec{a}} = l'_1 \vee \ldots \vee l'_m$ where, for each $i = 1, \ldots, m$,

$$l'_i = \begin{cases} p_i & \text{if } a_i = 0, \\ \neg p_i & \text{if } a_i = 1. \end{cases}$$

Since, for every $\vec{a} \in \{0,1\}^m$, $\neg\theta_{\vec{a}}$ is logically equivalent to $\gamma_{\vec{a}}$ it follows that $\bigwedge_{\vec{a} \in \mathbf{E}^c} \neg\theta_{\vec{a}}$ is logically equivalent to $\bigwedge_{\vec{a} \in \mathbf{E}^c} \gamma_{\vec{a}}$, in other words, $\neg\varphi^c$ is logically equivalent to $\psi := \bigwedge_{\vec{a} \in \mathbf{E}^c} \gamma_{\vec{a}}$ which is a CNF. As $\neg\varphi^c$ and ψ are logically equivalent it follows that ψ is also a complete description of $\mathbf{E}$. $\square$

Corollary 3.2.4 *For every $\varphi \in \mathsf{Prop}(\mathbf{P})$ there is a DNF which is logically equivalent to φ and there is a CNF which is logically equivalent to φ.*

Proof. Let $\mathbf{E}$ be the set of all $\vec{a} \in \{0,1\}^m$ such that $\vec{a} \models \varphi$. By Proposition 3.2.3, there is a DNF which is a complete description of $\mathbf{E}$ and there is a CNF which is a complete description of $\mathbf{E}$. Both the DNF and the CNF is logically equivalent to φ, as all three formulas are satisfied by exactly the same truth assignments. $\square$

The proof of Proposition 3.2.3 gives an algorithm for constructing a DNF $\bigvee_{\vec{a} \in \mathbf{E}} \theta_{\vec{a}}$ and a CNF $\bigwedge_{\vec{a} \in \mathbf{E}^c} \gamma_{\vec{a}}$ (both described in the proof) for a set of examples $\mathbf{E} \subseteq \{0,1\}^m$. Let

$\mathbf{E}^c = \{0,1\}^m \setminus \mathbf{E}$; in other words, $\mathbf{E}^c$ is the set of truth assignments which do not belong to $\mathbf{E}$ and we call $\mathbf{E}^c$ the *complement of* $\mathbf{E}$.

Since the DNF $\bigvee_{\vec{a} \in \mathbf{E}} \theta_{\vec{a}}$ contains one disjunct for every $\vec{a} \in \mathbf{E}$ and all disjuncts have length at most $3m$ it follows that the DNF can be constructed in linear time (and space) in $|\mathbf{E}|$ if we keep m constant. With the "big O" notation it means that the time (and space) needed for constructing the DNF is $O(|\mathbf{E}|)$. Since the CNF $\bigwedge_{\vec{a} \in \mathbf{E}^c} \gamma_{\vec{a}}$ contains one conjunct for every $\vec{a} \in \mathbf{E}^c$ and all conjuncts have length at most $3m$ it follows that the CNF can be constructed in linear time (and space) in $|\mathbf{E}^c|$, that is in time (and space) $O(|\mathbf{E}^c|)$.

However, note that $|\{0,1\}^m| = 2^m$, so the number of truth assignments grows exponentially with m which denotes the number of propositional variables. In a real application m may be large enough, say $m = 1000$, that 2^m is a huge number. In comparison to 2^m the set of examples $\mathbf{E} \subseteq \{0,1\}^m$ is typically going to be "tiny". For example, if $m = 1000$ and $|\mathbf{E}| = 10^9$ then the quotient $10^9/2^{1000}$ is a miniscule number and the difference $2^{1000} - 10^9$ is still a huge number, for practical purposes not much less that 2^{1000}. In the context of this section we are mainly interested in deriving a CNF (a logic program) from $\mathbf{E}$ and the CNF constructed in the proof of Proposition 3.2.3 is $\bigwedge_{\vec{a} \in \mathbf{E}^c} \gamma_{\vec{a}}$ which has $|E^c| = 2^{1000} - 10^9$ conjucts (or clauses) if $m = 1000$ and $|\mathbf{E}| = 10^9$. Somewhat ironically, if the number of examples $|\mathbf{E}|$ is close to 2^{1000} then $|\mathbf{E}^c|$ may be "small enough" that the size of the CNF is of practically managable size. However, already 2^{300} is larger than estimates of the number of atoms in the observable universe, so having example sets of size larger 2^{300} seems rather unrealistic.

To my knowledge no known algorithm solves the following problem within polynomial time in $|\mathbf{E}|$: given arbitrary m and arbitrary $\mathbf{E} \subseteq \{0,1\}^m$ find a CNF which is a complete description of $\mathbf{E}$. There is, however, a related problem which may, in many real applications, be easier to deal with computationally.

Definition 3.2.5 Let $\mathbf{P} = \{p_1, \ldots, p_m\}$ be a set of propositional variables and let $\mathbf{E} \subseteq \{0,1\}^m$ be a set of truth assignments.
(a) We say that a formula φ *covers* $\mathbf{E}$ if $\vec{a} \models \varphi$ for every $\vec{a} \in \mathbf{E}$.
(b) We say that a formula φ *avoids* $\mathbf{E}$ if $\vec{a} \models \neg\varphi$ for every $\vec{a} \in \mathbf{E}$.

Suppose that we are given $\mathbf{E} \subseteq \{0,1\}^m$ and $\mathbf{N} \subseteq \{0,1\}^m$ such that $\mathbf{E}$ and $\mathbf{N}$ are disjoint (i.e. $\mathbf{E} \cap \mathbf{N} = \emptyset$). Let us consider $\mathbf{E}$ as a set of "positive" examples and $\mathbf{N}$ as a set of "negative" examples. If φ covers $\mathbf{E}$ *and* avoids $\mathbf{N}$, then all positive examples satisfy φ and no negative example satisfies φ. It follows directly from the definitions that if $\mathbf{E} \subseteq \{0,1\}^m$ and φ is a complete description of $\mathbf{E}$, then φ covers $\mathbf{E}$ and avoids $\mathbf{E}^c = \{0,1\}^m \setminus \mathbf{E}$. But if $\mathbf{N} \neq \mathbf{E}^c$ then there is φ which covers $\mathbf{E}$ and avoids $\mathbf{N}$, but φ is not a complete description of $\mathbf{E}$. Thus the problem of finding φ which covers $\mathbf{E}$ and avoids $\mathbf{N}$ can be seen as a weakening, or an approxiamtion, of the problem of finding φ which is a complete description of $\mathbf{E}$. In the context of logic programs we are in addition interested in finding a CNF (that is, a logic program) with the mentioned properties.

Proposition 3.2.6 *Let $\mathbf{E}, \mathbf{N} \subseteq \{0,1\}^m$ be disjoint. For constant m one can find a CNF ψ which covers $\mathbf{E}$ and avoids $\mathbf{N}$ in linear time in $|\mathbf{N}|$.*

Proof. Suppose that $\mathbf{E}, \mathbf{N} \subseteq \{0,1\}^m$ are disjoint. We use the idea from the proof of Proposition 3.2.3. For every $\vec{a} = (a_1, \ldots, a_m) \in \{0,1\}^m$ let $\gamma_{\vec{a}} = l'_1 \vee \ldots \vee l'_m$ where, for each $i = 1, \ldots, m$,

$$l'_i = \begin{cases} p_i & \text{if } a_i = 0, \\ \neg p_i & \text{if } a_i = 1. \end{cases}$$

Then let $\psi = \bigwedge_{\vec{a} \in \mathbf{N}} \gamma_{\vec{a}}$. Observe that the number of disjuncts (or clauses) in ψ is equal to $|\mathbf{N}|$, so the length of ψ is linear in $|\mathbf{N}|$ assuming that m is constant (or bounded from above). By arguing as in the proof of Proposition 3.2.3 it follows that ψ is a complete description of $\{0,1\}^m \setminus \mathbf{N}$. Since $\mathbf{E} \subseteq \{0,1\} \setminus \mathbf{N}$ it follows that ψ covers $\mathbf{E}$ and avoids $\mathbf{N}$. $\square$

Suppose that $m = 1000$ as in the discussion earlier in this section, so $|\{0,1\}^m| = 2^{1000}$. In a real application when we want to find a CNF which covers $\mathbf{E} \subseteq \{0,1\}^m$ and avoids $\mathbf{N} \subseteq \{0,1\}^m$ both $|\mathbf{E}|$ and $|\mathbf{N}|$ are likely (or perhaps necessarily) to be much smaller than 2^{1000}. Perhaps we have something like $|\mathbf{E}|, |\mathbf{N}| \leq 10^9$. Although 10^9 may be considered a large number a search that takes time 10^9 is managable by computers today, but one cannot say that about a search that takes time 2^{1000}. The approach to only consider sets of positive examples $\mathbf{E}$ and negative examples $\mathbf{N}$ of managable size could be defended on the ground that (in real applications) it may be the case that the vast majority of the 2^{1000} (if $m = 1000$) mathematically possible truth assignments are not realized in the world we live in, due to various constraints (such as physical laws for example).

3.3 More on algorithmic issues

In this section (as in the previous) we consider propositional logic. The algorithms for finding a CNF with the desired property that can be derived from the proofs of propositions 3.2.3 and 3.2.6 are quite “brute force” and considers all truth assignments in $\{0,1\}^m \setminus \mathbf{E}$ or in $\mathbf{N}$, respectively. These algorithms often produce unnecessarily complicated CNF's. More sophisticated algorithms exist which “refines” an approximation to the CNF one wants to find, step by step, until the desired CNF has been found. Before describing such an algorithm we introduce some notation, terminology and conventions.

We let $\top$ denote a clause which is always true (such as $p \vee \neg p$). Let $\bigwedge_{i=n}^{n} \gamma_i$ be a CNF, so each γ_i is a clause (a disjunction of literals). *In computer science, a CNF $\bigwedge_{i=n}^{n} \gamma_i$, is often represented as a set $H = \{\gamma_1, \ldots, \gamma_n\}$ of clauses. We adopt this convention in this section.*

Let $\mathbf{P} = \{p_1, \ldots, p_m\}$ be a set of propositional variables. For all $\varphi, \psi \in \mathsf{Prop}(\mathbf{P})$ we define

$$\varphi \preccurlyeq \psi \text{ if and only if } \varphi \models \psi.$$

(Recall that $\varphi \models \psi$ means that for every truth assignment $\vec{a}$, if $\vec{a} \models \varphi$ then $\vec{a} \models \psi$.) Then $\preccurlyeq$ is a *preorder* on $\mathsf{Prop}(\mathbf{P})$, meaning that

- $\varphi \preccurlyeq \varphi$ for all $\varphi \in \mathsf{Prop}(\mathbf{P})$ and
- if $\varphi, \psi, \chi \in \mathsf{Prop}(\mathbf{P})$, $\varphi \preccurlyeq \psi$ and $\psi \preccurlyeq \chi$, then $\varphi \preccurlyeq \chi$.

Let $\mathbf{C}$ be the set of all clauses (in $\mathsf{Prop}(\mathbf{P})$) up to logical equivalence; by this I mean that for every clause $\gamma \in \mathsf{Prop}(\mathbf{P})$ there is exactly one clause $\gamma' \in \mathbf{C}$ such that γ and γ' are logically equivalent. The restriction of $\preccurlyeq$ to $\mathbf{C}$ has the following additional property:

- If $\varphi, \psi \in \mathbf{C}$, $\varphi \preccurlyeq \psi$ and $\psi \preccurlyeq \varphi$, then $\varphi = \psi$.

This means that $\preccurlyeq$ restricted to $\mathbf{C}$ is a *partial order* (in fact, it is even a *lattice*). It is a well known result that every partial order can be extended to a *linear order* (also called *total order*). In the present context this means that there is a linear order $\sqsubseteq$ on $\mathbf{C}$ such that

- (extending $\preccurlyeq$) for all $\varphi, \psi \in \mathbf{C}$, if $\varphi \preccurlyeq \psi$ then $\varphi \sqsubseteq \psi$, and

- (linearity) for all $\varphi, \psi \in \mathbf{C}$, $\varphi \sqsubseteq \psi$ or $\psi \sqsubseteq \varphi$.

Now we are ready to describe two versions of an algorithm which, given $\mathbf{E}, \mathbf{N} \subseteq \{0,1\}^m$ such that $\mathbf{E} \cap \mathbf{N} = \emptyset$ and $\mathbf{N} \neq \emptyset$ returns a CNF H (in set form) which covers $\mathbf{E}$ and avoids $\mathbf{N}$. The idea is that we start with $H := \{\top\}$ which means that H covers any set of truth assignments, so in particular it covers $\mathbf{E}$. Initially we set $\mathbf{Q} := \mathbf{C}$. Along the whole procedure we make sure that H always covers $\mathbf{E}$. The aim is to eventually also make H avoid $\mathbf{N}$. To reach this goal H is refined in each step by adding a "minimally restrictive" or "maximally restrictive" (depending on the version of the algorithm) clause $\gamma \in \mathbf{Q}$ to H such that $H \cup \{\gamma\}$ still covers $\mathbf{E}$ and then we remove γ from $\mathbf{Q}$.

Algorithm 1; producing a maximally general set of clauses H which covers $\mathbf{E}$ and avoids $\mathbf{N}$:

```
Input: $\mathbf{E}, \mathbf{N} \subseteq \{0,1\}^m$ such that $\mathbf{E} \cap \mathbf{N} = \emptyset$.

    $H := \{\top\}$. (Then $H$ covers $\mathbf{E}$.)
    $\mathbf{Q} := \mathbf{C}$.
    As long as $H$ does not avoid $\mathbf{N}$ do:
        Find the largest (with respect to $\sqsubseteq$) $\gamma \in \mathbf{Q}$ such that $H \cup \{\gamma\}$ covers $\mathbf{E}$ and
        $H \not\models H \cup \{\gamma\}$. Set $H := H \cup \{\gamma\}$ and $\mathbf{Q} := \mathbf{Q} \setminus \{\gamma\}$.

Output: $H$ (which covers $\mathbf{E}$ and avoids $\mathbf{N}$).
```

Algorithm 2; producing a possibly more specific (less general) set of clauses H which covers $\mathbf{E}$ and avoids $\mathbf{N}$:

Just as Algorithm 1 *except* that in the loop we replace "*largest*" by "*smallest*".

The central step in each of the algorithms is justified by the following:

Lemma 3.3.1 *Suppose that a set of clauses (a CNF) H covers $\mathbf{E} \subseteq \{0,1\}^m$ but does not avoid $\mathbf{N} \subseteq \{0,1\}^m$, where $\mathbf{E} \cap \mathbf{N} = \emptyset$. Then there is a clause γ such that $H \cup \{\gamma\}$ covers $\mathbf{E}$ and $H \not\models H \cup \{\gamma\}$ (so $H \cup \{\gamma\}$ is more restrictive than H).*

Proof. If H does not avoid $\mathbf{N}$, then there is $\vec{a} \in \mathbf{N}$ such that $\vec{a} \models H$. Let $\gamma_{\vec{a}}$ be the clause defined in the proof of Proposition 3.2.6. Then $\gamma_{\vec{a}}$ is satisfied by every truth assignment other than $\vec{a}$, so in particular it is satisfied by every truth assignment in $\mathbf{E}$. So if H covers $\mathbf{E}$ then $H \cup \{\gamma_{\vec{a}}\}$ also covers $\mathbf{E}$. By the choice of $\vec{a}$ we have $\vec{a} \models H$ and by the definition of $\gamma_{\vec{a}}$ we have $\vec{a} \not\models \gamma_{\vec{a}}$. Thus $H \not\models H \cup \{\gamma_\gamma\}$. □

Remark 3.3.2 (a) Before each new iteration of the main step of either version of the algorithm, H will avoid some $\mathbf{N}' \subseteq \mathbf{N}$. If $\mathbf{N}' \neq \mathbf{N}$ then, after the next iteration, Algorithm 1 produces $H := H \cup \{\gamma\}$ for some clause γ such that the new H avoids some $\mathbf{N}''$ such that $\mathbf{N}' \subset \mathbf{N}'' \subseteq \mathbf{N}$ such that $|\mathbf{N}''| = |\mathbf{N}'| + 1$. Consequently, Algorithm 1 will always make $|\mathbf{N}|$ iterations. If $\mathbf{N}' \neq \mathbf{N}$ then, after the next iteration, Algorithm 2 produces $H := H \cup \{\gamma\}$ for some clause γ such that the new H avoids some $\mathbf{N}''$ such that $\mathbf{N}' \subset \mathbf{N}'' \subseteq \mathbf{N}$ such that $|\mathbf{N}''| \geq |\mathbf{N}'| + 1$ where the inequality may be strict ($>$). This means that Algorithm 2 may need fewer than $|\mathbf{N}|$ iterations to complete its task. If γ and γ' are clauses and $\gamma \sqsubset \gamma'$, then γ has fewer literals than γ'. Consequently, Algorithm 2 may produce a set of simpler clauses than Algorithm 1.

(b) Algorithm 1 produces a set of clauses H which is equivalent to the formula ψ produced in the proof of Proposition 3.2.6. This means that for all $\vec{a} \in \{0,1\}^m$, $\vec{a} \models H$ if and only if $\vec{a} \notin \mathbf{N}$, so H in fact covers the complement of $\mathbf{N}$ which may be much larger than $\mathbf{E}$. Algorithm 2 produces a set of clauses H which covers some $\mathbf{E}' \supseteq \mathbf{E}$ such that $\mathbf{E}' \cap \mathbf{N} = \emptyset$ and avoids $\mathbf{N}$, but here $\mathbf{E}'$ need not be equal to the complement of $\mathbf{N}$.

Remark 3.3.3 The approach described above to finding a logic program from a set of postivie and negative examples is called *learning from interpretations*. There is also an approach called *learning from entailment*; see for example [1, 3, 16].

3.4 First-order logic programs

In this section we consider first-order logic and it will be convenient to denote a sequence of variables by $\vec{x}$ (or $\vec{y}$ etc). So if $\varphi(\vec{x})$ denotes a formula then it is assumed that all free variables of φ occur in the sequence $\vec{x}$, but we do not insist that every variable in the sequence $\vec{x}$ occurs in the formula denoted by $\varphi(\vec{x})$. For example, if R is a binary relation symbol and $\vec{x} = (x_1, x_2, x_3, x_4)$ then the expression $\varphi(\vec{x})$ could denote the formula $R(x_1, x_3)$.

The ideas from Section 3.1 make sense for first-order logic, but I nevertheless begin by recalling them with notation more adapted to first-order logic. Recall Definition 1.1.2 for terminology. By a (first-order) logic program we mean a set of clauses. Recall that a clause $\theta_1(\vec{x}) \vee \ldots \vee \theta_n(\vec{x})$ is, for any $1 \leq k < n$, logically equivalent to

$$(\neg\theta_1(\vec{x}) \wedge \ldots \wedge \neg\theta_k(\vec{x})) \rightarrow (\theta_{k+1}(\vec{x}) \vee \ldots \vee \theta_n(\vec{x}))$$

and each $\neg\theta_i(\vec{x})$ is logically equivalent to a literal. Moreover, a formula of the form

$$(\varphi_1(\vec{x}) \wedge \ldots \wedge \varphi_n(\vec{x})) \rightarrow (\psi_1(\vec{x}) \vee \ldots \vee \psi_m(\vec{x})) \tag{3.4.1}$$

can be viewed as a "rule" for making new conclusions. That is, if we know that all $\varphi_1(\vec{x}), \ldots, \varphi_n(\vec{x}))$ hold, then (3.4.1) expresses that we can draw the conclusion that at least one of $\psi_1(\vec{x}), \ldots, \psi_m(\vec{x})$ holds. Alternatively viewed, if we know that at least one of $\psi_1(\vec{x}), \ldots, \psi_m(\vec{x})$ holds, then (3.4.1) expresses that a possible cause of this is that all $\varphi_1(\vec{x}), \ldots, \varphi_n(\vec{x}))$ hold.

We now make the above informal argument more precise which involves universally quantifying the variables in $\vec{x} = (x_1, \ldots, x_s)$. Suppose that we work with a first-order language L and that $\mathcal{A}$ is an L-structure. Also assume that for every $a \in A$ there is a constant symbol $\hat{a}$ which is interpreted as a in $\mathcal{A}$. Finally suppose that

$$\mathcal{A} \models \forall\vec{x}\big((\varphi_1(\vec{x}) \wedge \ldots \wedge \varphi_n(\vec{x})) \rightarrow (\psi_1(\vec{x}) \vee \ldots \vee \psi_m(\vec{x}))\big) \tag{3.4.2}$$

where '$\forall\vec{x}$' is an abbreviation of '$\forall x_1 \forall x_2 \ldots \forall x_s$'. It follows that if $a_1, \ldots, a_s \in A$ and $\mathcal{A} \models \bigwedge_{i=1}^{n} \varphi_1(\hat{a}_1, \ldots, \hat{a}_s)$, then $\mathcal{A} \models \bigvee_{i=1}^{m} \psi(\hat{a}_1, \ldots, \hat{a}_s)$. Thus, provided that (3.4.2) holds, we can use the formula (3.4.1) as a rule which applies to all elements in the domain of the structure $\mathcal{A}$. Observe that if $\mathcal{A} \models \neg\exists\vec{x}(\varphi_1(\vec{x}) \wedge \ldots \wedge \varphi_n(\vec{x}))$, where '$\exists\vec{x}$' abbreviates '$\exists x_1 \exists x_2 \ldots \exists x_3$', then (3.4.2) holds, but in this case the the rule (3.4.1) is uninteresting, at least for the structure $\mathcal{A}$, because the conjunction of conditions $\varphi_1(\vec{x}) \wedge \ldots \wedge \varphi_n(\vec{x})$ is not satisfied by any elements from the domain of $\mathcal{A}$.

Now we turn to the algorithmic problem of finding literals $\varphi_1(\vec{x}), \ldots, \varphi_n(\vec{x}))$, $\psi_1(\vec{x}), \ldots, \psi_m(\vec{x})$ such that (3.4.2) holds for a given structure $\mathcal{A}$, where $\mathcal{A}$ can be viewed as the example from which we learn rules. Alternatively we can start with a set $\mathbf{E}$ of example structures and look for literals such that (3.4.2) holds for every structure $\mathcal{A} \in \mathbf{E}$. Note that all which has been said in the previous paragraph makes sense without the assumption that the formulas $\varphi_i(\vec{x})$, $i = 1, \ldots, n$, and $\psi_j(\vec{x})$, $j = 1, \ldots, m$, are literals. But the algorithmic complexity increases very fast if we allow formulas with quantifiers, which is one reason why one usually restricts attention to formulas like (3.4.1) where all $\varphi_i(\vec{x})$ and $\psi_i(\vec{x})$ are literals. Moreover, this often works well in practical applications. Now we state the learning problem that we consider until Section 3.4.3.

Problem 3.4.1 (Learning rules) Let L be first-order language and let $\mathbf{E}$ be a set of L-structures (the examples). Find literals $\varphi_1(\vec{x}), \ldots, \varphi_n(\vec{x}))$, $\psi_1(\vec{x}), \ldots, \psi_m(\vec{x})$ such that (3.4.2) holds for every $\mathcal{A} \in \mathbf{E}$ and such that $\mathcal{A} \models \exists\vec{x}(\varphi_1(\vec{x}) \wedge \ldots \wedge \varphi_n(\vec{x}))$ for at least one $\mathcal{A} \in \mathbf{E}$. A common special case is when $m = 1$ and the literal $\psi_1(\vec{x})$ expresses some property/relationship of interest, and we want to find literals $\varphi_1(\vec{x}), \ldots, \varphi_n(\vec{x}))$ such that (3.4.2) holds for every $\mathcal{A} \in \mathbf{E}$.

Warning: To find rules in the above sense the sequence one or more of the literals $\varphi_1(\vec{x}), \ldots,$ $\varphi_n(\vec{x}))$ may need to contain variables that do not occur in any of the literals $\psi_1(\vec{x}), \ldots, \psi_m(\vec{x})$; in other words, some variable in the sequence $\vec{x}$ may not occur in any of the variables denoted by $\psi_1(\vec{x}), \ldots, \psi_m(\vec{x})$. (This is illustrated by the example below.)

Problem 3.4.1 is essentially a search problem. We first look at an example and then consider a general method.

3.4.1 Example: Learning the concepts of daughter and grandmother

In this example data is given about some relationships between some individuals. The data contains information about the sex of each individual and for every pair a and b of individuals it contains information about whether a is a parent of b, whether a is a daughter of b, and whether a is the grandmother of b. The first task is to find out whether the relationship of an individual being the daughter of an individual can be inferred from other information, based on the given data. We then consider the same problem for the relation of "grandmothership". The data can be formally described by an L-structure for a suitable language L. Let the set of individuals about whom we have information be denoted a, b, e, g, h, m, n and t. Let F, P, D and G be relation symbols where F is unary and the other ones binary and let L contain these relation symbols. We interpret the formulas $F(x)$, $P(x, y)$, $D(x, y)$ and $G(x, y)$ as expressing that "x is female", "x is a parent of y", "y is a daughter of x", and "x is a grandmother of y", respectively. Suppose that the information about the individuals a, b, e, g, h, m, n and t is described by an L-structure $\mathcal{A}$ where

- $A = \{a, b, e, g, h, m, n, t\}$ is the domain of $\mathcal{A}$,
- $F^{\mathcal{A}} = \{m, h, e, n\}$ (i.e. m, n, e, and n are females),
- $P^{\mathcal{A}} = \{(g, m), (h, m), (h, t), (t, e), (n, e), (a, t), (e, b)\}$ (i.e. g is a parent of m, h is a parent of m, and so on),
- $D^{\mathcal{A}} = \{(g, m), (h, m), (t, e), (n, e)\}$, and
- $G^{\mathcal{A}} = \{(h, e), (n, b)\}$.

Also assume that L contains constant symbols $\hat{a}$, $\hat{b}$, $\hat{e}$, etc which are interpreted as a, b, e etc. If $\mathbf{E} = \{\mathcal{A}\}$ then our problem is as stated in Problem 3.4.1. We fix two variables x

and y and want to find a sequence of variables $\vec{x}$ that contains x and y (and possibly more variables) and literals $\varphi_1(\vec{x}), \ldots, \varphi_n(\vec{x})$ such that

$$\mathcal{A} \models \forall \vec{x}\big((\varphi_1(\vec{x}), \ldots, \varphi_n(\vec{x})) \to D(x, y)\big).$$

In our initial attempt we can let $\vec{x} = (x, y)$ (so no extra variables are considered). Then test, for example, if $\mathcal{A} \models \forall x \forall y\big(P(x, y) \to D(x, y)\big)$. This is false because, for example, $\mathcal{A} \models P(\hat{h}, \hat{t}) \wedge \neg D(\hat{h}, \hat{t})$ (that is, h is a parent of t but t is not a daughter of h). Since $P(x, y)$ did not imply $D(x, y)$ for all instantiations of individuals from $\mathcal{A}$ for the variables x and y, we can try to strengthen the assumptions (the literals to the left of $\to$). What about $\forall x \forall y\big((P(x, y) \wedge F(x)) \to D(x, y)\big)$? This formula is also false in $\mathcal{A}$ because $\mathcal{A} \models P(\hat{e}, \hat{b}) \wedge F(\hat{e}) \wedge \neg D(\hat{e}, \hat{b})$. What if we strengthen the assumptions further by considering $\forall x \forall y\big((P(x, y) \wedge F(x) \wedge F(y)) \to D(x, y)\big)$? One can easily check that we do have $\mathcal{A} \models \forall x \forall y\big((P(x, y) \wedge F(x) \wedge F(y)) \to D(x, y)\big)$ and $\mathcal{A} \models \exists x \exists y\big((P(x, y) \wedge F(x) \wedge F(y)\big)$. But are the assumptions unnecessarily strong? One can check that $\mathcal{A} \models \forall x \forall y\big((P(x, y) \wedge F(y)) \to D(x, y)\big)$ which is better if we want the conditions to the left of '$\to$' to explain as "accurately as possible" when $D(x, y)$ holds, that is, when y is the daughter of x, given the evidence encoded by $\mathcal{A}$. In fact the last formula is optimal in the sense that we have $\mathcal{A} \models \forall x \forall y\big((P(x, y) \wedge F(y)) \leftrightarrow D(x, y)\big)$. So we have found the ordinary definition of when y is a daughter of x, which is that x is a parent of y and y is female.

Now we consider the problem of finding, based on the evidence encoded by $\mathcal{A}$, sufficient (and possibly even necessary) conditions for x to be the grandmother of y. It turns out (as the reader can verify) that there does not exist literals $\varphi_1(x, y), \ldots, \varphi_n(x, y)$ such that none of the literals contain the symbol G (for grandmother), $\mathcal{A} \models \forall x \forall y\big((\varphi_1(x, y) \wedge \ldots \wedge \varphi_n(x, y)) \to G(x, y)\big)$ and $\mathcal{A} \models \exists x \exists y\big(\varphi_1(x, y) \wedge \ldots \wedge \varphi_n(x, y)\big)$. Thus we need to allow more variables in the literals to the left of the symbol '$\to$', so we try with x, y, z. By trying different combinations of literals we can see that

$$\begin{aligned}
&\mathcal{A} \models \forall x \forall y \forall z\big(((P(x, z) \wedge P(z, y) \wedge F(x)) \to G(x, y)\big) \quad \text{and} \\
&\mathcal{A} \models \exists x \exists y \exists z((P(x, z) \wedge P(z, y) \wedge F(x)).
\end{aligned}$$

Moreover, the converse implication holds in $\mathcal{A}$ as well, so we have

$$\mathcal{A} \models \forall x \forall y \forall z\big(((P(x, z) \wedge P(z, y) \wedge F(x)) \leftrightarrow G(x, y)\big).$$

3.4.2 A systematic search procedure

Suppose that the language L has no function symbols and only finitely many relation and constant symbols. It follows that for every sequence $\vec{x}$ of variables there are only finitely many literals such that every variable in the literal occurs in $\vec{x}$. Let $\mathbf{E}$ be a finite set of finite L-structures.

Fix a disjunction of literals ψ. We search for conditions which imply ψ. Fix a sequence of variables $\vec{x}$ such that every variable in ψ occurs in $\vec{x}$. We will search for rules of the form $\forall \vec{x}(\varphi(\vec{x}) \to \psi(\vec{x}))$ where $\varphi(\vec{x})$ is a conjunction of literals.

Let $Q_{\vec{x}}$ be the (finite) set of all *atomic* L-formulas such that every variable in the formula occurs in $\vec{x}$ and the relation symbol in the formula does not appear in $\psi(\vec{x})$. Let $\chi_1(\vec{x}), \ldots, \chi_t(\vec{x})$ be a list of all formulas in $Q_{\vec{x}}$. For each $\chi_i(\vec{x})$ let $\theta_i(\vec{x})$ either be the same as $\chi_i(\vec{x})$ or let it be $\neg\chi_i(\vec{x})$ (so we make a choice for each $i = 1, \ldots, t$). Thus $\theta_1(\vec{x}), \ldots, \theta_t(\vec{x})$ is a list of literals.

(1) Initially set $\varphi(\vec{x}) := \top$ and $k := 1$ (where $\top$ has no variable and is interpreted as true in every structure).

(2) (a) If $\mathcal{A} \models \forall\vec{x}(\varphi(\vec{x}) \to \psi(\vec{x}))$ for every $\mathcal{A} \in \mathbf{E}$ and $\mathcal{A} \models \exists\vec{x}\varphi(\vec{x})$ for at least one $\mathcal{A} \in \mathbf{E}$, then we are finished and return the rule $\varphi(\vec{x}) \to \psi(\vec{x})$.

(b) If the condition in (a) does not hold and $k \leq t$, then set $\varphi(\vec{x}) := \varphi(\vec{x}) \wedge \theta_k(\vec{x})$ and $k := k + 1$ and then return to (a).

(c) If the condition in (a) does not hold and $k > t$, then we are finished and conclude that when using the list $\theta_1(\vec{x}), \dots, \theta_t(\vec{x})$ of literals the search of a rule failed.

When the step (2) has been iterated until termination we can choose a new list of literals $\theta_1(\vec{x}), \dots, \theta_t(\vec{x})$ in the way explained before step (1) and then repeat (1) and (2). Suppose that after we have iterated this procedure as many times as we like, we have found $\varphi_1(\vec{x}), \dots, \varphi_n(\vec{x})$ such that, for every $i = 1, \dots, n$, $\varphi_i(\vec{x})$ is a conjunction of literals, $\mathcal{A} \models \forall\vec{x}(\varphi_i(\vec{x}) \to \psi(\vec{x}))$ for every $\mathcal{A} \in \mathbf{E}$, and $\mathcal{A} \models \exists\vec{x}\varphi_i(\vec{x})$ for at least one $\mathcal{A} \in \mathbf{E}$. So if $n > 1$ we have found several rules. If for some $i \neq j$ we have that

$$\mathcal{A} \models \forall\vec{x}(\varphi_i(\vec{x}) \to \varphi_j(\vec{x})) \quad \text{and} \quad \mathcal{A} \not\models \forall\vec{x}(\varphi_j(\vec{x}) \to \varphi_i(\vec{x}))$$

for all $\mathcal{A} \in \mathbf{E}$, then the assumption $\varphi_i(\vec{x})$ is unnecessarily strong. This can be a motivation to keep the rule $\varphi_j(\vec{x}) \to \psi(\vec{x})$ and discard $\varphi_i(\vec{x}) \to \psi(\vec{x})$. But on the other hand, if $\varphi_i(\vec{x})$ is a syntactically simpler formula (e.g. has shorter length as a string of symbols) than $\varphi_j(\vec{x})$, then one may prefer the rule $\varphi_i(\vec{x}) \to \psi(\vec{x})$, because simpler formulas/rules are less time consuming to handle computationally. By using some "scoring system" it is possible to discard assumptions (i.e. conjunctions of literals) which are unnecessarily strong and/or assumptions which are very complicated.

3.4.3 More about concept learning

The issues discussed earlier in Section 3.4 can be seen as a sort of "concept learning", because we searched for "rules" (motivated by examples) which could be thought of as explaining a concept. In this brief section we describe another variant of concept learning. The idea is that we are given (as in Sections 3.2 and 3.3) a set of examples and a set of nonexamples and the task is to distinguish the examples from the nonexamples. In the present context we look for a first-order formula which distinguishes the examples from the nonexamples, that is, the formula should be true in all examples, viewed as first-order structures, and false in all nonexamples.

More precisely, we let L be a first-order language, $\mathbf{E}$ (the set of examples) a set of L-structures and $\mathbf{N}$ (the set of nonexamples) a set of L-structures. *The problem is to find an L-sentence (i.e. L-formula without free variables) φ such that $\mathcal{A} \models \varphi$ for all $\mathcal{A} \in \mathbf{E}$ and $\mathcal{A} \models \neg\varphi$ for all $\mathcal{A} \in \mathbf{N}$.* In general this problem is undecidable due to results from Section 2.1, even if all structures in $\mathbf{E}$ and in $\mathbf{N}$ are finite, which they will of course be in practical situations. Moreover, for this problem to have a solution, even theoretically, we must assume that for every $\mathcal{A} \in \mathbf{E}$ there is no $\mathcal{B} \in \mathbf{N}$ such that $\mathcal{A}$ and $\mathcal{B}$ are isomorphic, because if they are isomorphic then we have $\mathcal{A} \models \psi$ if and only if $\mathcal{B} \models \psi$ for *all* L-sentences ψ. See for example [10] for a definition of *isomorphism* in the context of first-order structures, but intuitively two structures are isomorphic if they are the same except (possibly) for the "names" of the elements in their domains. (Isomorphism between for example graphs or groups are just special cases of the concept of isomorphism between first-order structures.)

In practical applications the sets $\mathbf{E}$ and $\mathbf{N}$ will be finite, as well as their members. In this situation the problem has a theoretical solution, as stated below.

Theorem 3.4.2 *Let L be a first-order language with only finitely many relation, constant and function symbols. Suppose that $\mathbf{E}$ and $\mathbf{N}$ are finite sets of finite L-structures and assume that for every $\mathcal{A} \in \mathbf{E}$ there is no $\mathcal{B}$ such that $\mathcal{A}$ and $\mathcal{B}$ are isomorphic. Then there is a sentence φ such that $\mathcal{A} \models \varphi$ for all $\mathcal{A} \in \mathbf{E}$ and $\mathcal{B} \models \neg\varphi$ for all $\mathcal{B} \in \mathbf{N}$. Moreover, φ can be taken to be a disjunction of $|\mathbf{E}|$ formulas of the form $\exists\vec{x}\psi(\vec{x}) \wedge \neg\exists\vec{y}\theta(\vec{y})$ where $\psi(\vec{x})$ and $\theta(\vec{y})$ are quantifier-free and, for some $\mathcal{A} \in \mathbf{E}$, $|\vec{x}| = |A|$, $\exists\vec{x}\psi(\vec{x})$ completely describes the interpretations of the symbols of L in $\mathcal{A}$ (up to permutations of the domain), and $\neg\exists\vec{y}\theta(\vec{y})$ expresses that the domain has at most $|A|$ elements, by saying (for $k = |A|$) that there does not exist $x_1, \ldots, x_{k+1}$ such that $x_i \neq x_j$ if $i \neq j$.*

Proof sketch. A well-known result in model theory is that for every *finite* L-structure $\mathcal{A}$ there is a sentence $\varphi_{\mathcal{A}}$ such that for every L-structure $\mathcal{B}$, $\mathcal{B}$ is isomorphic to $\mathcal{A}$ if and only if $\mathcal{B} \models \varphi_{\mathcal{A}}$. (See for example [10, Proposition 2.81].)[3] Moreover, the proof shows that such $\varphi_{\mathcal{A}}$ exists which has the form $\exists\vec{x}\psi(\vec{x}) \wedge \neg\exists\vec{y}\theta(\vec{y})$ where $\psi(\vec{x})$ and $\theta(\vec{y})$ are quantifier-free and the other conditions stated in the theorem hold. If we let $\varphi = \bigvee_{\mathcal{A}\in\mathbf{E}} \varphi_{\mathcal{A}}$, then $\mathcal{A} \models \varphi$ for all $\mathcal{A} \in \mathbf{E}$ and $\mathcal{B} \models \neg\varphi$ for all $\mathcal{B} \in \mathbf{N}$, because of the construction and the assumption that there does not exist $\mathcal{A} \in \mathbf{E}$ and $\mathcal{B} \in \mathbf{N}$ which are isomorphic. $\square$

Unfortunately the procedure for constructing φ as in Theorem 3.4.2 is computationally very time consuming and φ will be large if $\mathbf{E}$ contains some large structure or if it contains many structures. More precisely, φ is a disjunction of $|\mathbf{E}|$ formulas and if $\mathcal{A} \in \mathbf{E}$ then the disjunct corresponding to $\mathcal{A}$ will use $|A|$ variables (where A is the domain of $\mathcal{A}$). Thus if $\mathbf{E}$ is large or contains a large structure then the formula φ constructed in the proof of Theorem 3.4.2 will be very impractical to use.

Instead of searching for a complete description of all structures in $\mathbf{E}$ one can search for relatively small "patterns" that appear in all structures in $\mathbf{E}$ but not in any structure in $\mathbf{N}$. This is illustrated by the next example.

Example 3.4.3 Suppose that $\mathbf{E}$ and $\mathbf{N}$ contain unordered graphs, viewed as L-structures for a language L with a binary relation symbol R representing the edge relation. Also assume that every graph in $\mathbf{E}$ has a cycle of length at most 4 and that no graph in $\mathbf{N}$ has a cycle of length at most 4. Then every graph in $\mathbf{E}$ satisfies the sentence

$$\begin{aligned}\exists x_1\exists x_2\exists x_3\exists x_4\big(&\big(R(x_1,x_2) \wedge R(x_2,x_3) \wedge R(x_3,x_1)\big)\vee \\ &\big(R(x_1,x_2) \wedge R(x_2,x_3) \wedge R(x_3,x_4) \wedge R(x_4,x_1)\big)\big)\end{aligned}$$

but no graph in $\mathbf{E}$ satisfies the same sentence.

If we have some information about all structures that we investigate, both the examples and the nonexamples, then the problem of distinguishing the examples from the nonexamples may be simplified. For example, if we deal with unordered graphs, as in Example 3.4.3, and if we also now that the maximum degree of the graph $\mathcal{A}$ is at most 5, say, then the problem of finding, up to isomorphism, all subgraphs of cardinality k which can be embedded into $\mathcal{A}$ is simpler (assuming that k is much smaller than $|A|$) than the same problem for a graph with the same number of vertices in general.

[3]This result does *not* hold for *infinite* structures.

Chapter 4

Logic and probability

In this chapter we develop two frameworks for dealing with formal logic and probability theory in a mathematically precise way. Section 4.1 recalls some elementary probability theory as well as a definition of Bayesian network. Section 4.2 considers the "possible worlds approach" to dealing with logic and probability. Section 4.3 considers the "probabilities on domains approach", or "statistical information approach", to dealing with logic and probability. Both approaches will be used in Chapter 5 where we will see how to construct a probabilistic model, more precisely a Bayesian network, from a set of (possibly incomplete) data and then using the Bayesian network to predict the probability of some event described by a possibly complex first-order formula.

4.1 Basic probability theory

In this section we recall some elementary definitions and results from probability theory.

Definition 4.1.1 Let S be a set which is finite or countably infinite.
(a) A function $\mu : S \to \mathbb{R}$ (where $\mathbb{R}$ is the set of real numbers) is called a *(discrete) probability distribution* (on S) if $0 \leq \mu(s) \leq 1$ for all $s \in S$ and $\sum_{s \in S} \mu(s) = 1$. If $\mu : S \to \mathbb{R}$ is a (discrete) probability distribution the we call (S, μ) a *(discrete) probability space.* Since we only consider discrete probability distributions in this text we omit the word "discrete".
(b) Every probability distribution $\mu : S \to \mathbb{R}$ can be extended to a function, which we usually give the same name μ, from the set of all subsets of S to $\mathbb{R}$ as follows: If $X \subseteq S$ then $\mu(X) = \sum_{s \in X} \mu(s)$. We call such an extension, defined on the set of all subsets of S, a *probability measure* on S.

Certain basic results follow from this definition. Proofs can be found in various introductory texts on probability theory.

Proposition 4.1.2 (Basics of probability measures) *Suppose that S is a finite or countably infinite set and that $\mu : S \to \mathbb{R}$ is a probability distribution.*

1. *If $X \subseteq Y \subseteq S$ then $\mu(X) \leq \mu(Y)$.*

2. *$\mu(\emptyset) = 0$ (where $\emptyset$ denotes the empty set).*

3. *For every $X \subseteq S$, $0 \leq \mu(X) \leq 1$.*

4. *If $X, Y \subseteq S$ and $X \cap Y = \emptyset$ (that is, if X and Y mutually exclusive events), then $\mu(X \cup Y) = \mu(X) + \mu(Y)$.*

5. *For every $X \subseteq S$, $\mu(S \backslash X) = 1 - \mu(X)$. (That is, the probability of the complementary event to X is $1 - \mu(X)$.)*

For the rest of this section let $\mu : S \to \mathbb{R}$ be a probability distribution. If $X, Y \subseteq S$ then, intuitively, "the conditional probabillity of X given (that) Y (is the case)" is the probability that s (taken from S) belongs to X *if we already know that s belongs to Y*. The precise definition is as follows:

Definition 4.1.3 (a) The *conditional probability of X given (or assuming) Y*, denoted $\mu(X \mid Y)$, is defined as

$$\mu(X \mid Y) = \frac{\mu(X \cap Y)}{\mu(Y)}.$$

(b) We say that X and Y are *independent* if $\mu(X \mid Y) = \mu(X)$.

(c) We say that X and Y are *(conditionally) independent given (or assuming) Z* if $\mu(X \mid Y \cap Z) = \mu(X \mid Z)$.

From the definition it follows that

X and Y are independent if and only if $\mu(X \cap Y) = \mu(X) \cdot \mu(Y)$

and

X and Y are (conditionally) independent given Z if and only if $\mu(X \cap Y \mid Z) = \mu(X \mid Z) \cdot \mu(Y \mid Z)$.

It also follows that if $S = X_1 \cup \ldots \cup X_n$ and the sets $X_1, \ldots, X_n$ are *mutually disjoint* (i.e. $X_i \cap X_j = \emptyset$ for all choices of $i \neq j$), then, for every $Y \subseteq S$,

$$\mu(Y) = \sum_{i=1}^{n} \mu(Y \mid X_i) \cdot \mu(X_i).$$

Definition 4.1.4 A *binary random variable* is a function from S to $\{0, 1\}$, that is, a function with domain S and values in $\{0, 1\}$.

Note that to a binary random variable $f : S \to \{0, 1\}$ we can naturally associate two events, that is, subsets of S. The first subset is $f^0 := \{s \in S : f(s) = 0\}$ and the other subset is $f^1 := \{s \in S : f(s) = 1\}$. We will use the following notation if $f_1, \ldots, f_k, g_1, \ldots, g_n$ are binary random variables:

$\mu(f = 0) := \mu(f^0)$,

$\mu(f = 1) := \mu(f^1)$,

if $i_1, \ldots, i_k, j_1, \ldots, j_n \in \{0, 1\}$, then
$\mu(f_1 = i_1, \ldots, f_k = i_k \mid g_1 = j_1, \ldots, g_n = j_n) :=$
$\mu((f_1)^{i_1} \cap \ldots \cap (f_k)^{i_k} \mid (g_1)^{j_1} \cap \ldots \cap (g_n)^{j_n})$.

Now we consider independence over a *set* of binary random variables.

Definition 4.1.5 Suppose that $X, Y_1, \ldots, Y_m, Z_1, \ldots, Z_n$ are binary random variables (i.e. functions from S to $\{0,1\}$). We say that X *is conditionally independent from* $Y_1, \ldots, Y_m$ *over* $Z_1, \ldots, Z_n$ if, for *every* choice of $j_1, \ldots, j_m, i_1, \ldots, i_n \in \{0,1\}$,

$$\mu(X = 1 \mid Y_1 = j_1, \ldots, Y_m = j_m, Z_1 = i_1, \ldots, Z_n = i_n) =$$
$$\mu(X = 1 \mid Z_1 = i_1, \ldots, Z_n = i_n) \text{ (whenever both sides are defined).}$$

It easily follows that X is conditionally independent from $Y_1, \ldots, Y_m$ over $Z_1, \ldots, Z_n$ if and only if, for *every* choice of $j_1, \ldots, j_m, i_1, \ldots, i_n \in \{0,1\}$,

$$\mu(X = 1, Y_1 = j_1, \ldots, Y_m = j_m \mid Z_1 = i_1, \ldots, Z_n = i_n) =$$
$$\mu(X = 1 \mid Z_1 = i_1, \ldots, Z_n = i_n) \cdot \mu(Y_1 = j_1, \ldots, Y_m = j_m \mid Z_1 = i_1, \ldots, Z_n = i_n).$$

Given a probability distribution $\mu : S \to \mathbb{R}$, we can, for any $n > 1$, define a probability distribution on S^n (where S^n is the cartesian product of S with itself n times, that is, S^n is the set of ordered n-tuples, or vectors of length n, of elements from S) as follows:

$$\text{For every } \vec{s} = (s_1, \ldots, s_n) \in S^n, \ \mu^n(\vec{s}) = \mu(s_1) \cdot \ldots \cdot \mu(s_n).$$

One can verify that μ^n is indeed a probability distribution on S^n. We may call μ^n the *product measure (on* S^n*) derived* from μ.

Definition 4.1.6 (Bayesian network) A *Bayesian network* is determined by the following data:

1. A probability space (S, μ).
2. A set $V = \{X_1, \ldots, X_n\}$ of binary random variables on S.
3. A directed acyclic graph (abbreviated DAG) $\mathcal{G}$ with vertex set V such that if there is an arrow (directed arc) from X_i to X_j then $i < j$.
4. To each vertex (binary random variable) $X_i \in V$ conditional probabilities are associated in such a way that the following holds:
   (a) For each j the set of parents of X_j, denoted $Par(X_j)$, is minimal with the property that for every $i < j$, X_i and X_j are independent over $Par(X_j)$.[1]
   (b) The joint probability distribution on $X_1, \ldots, X_n$ induced by μ is determined by the conditional probabilities associated with the vertices of $\mathcal{G}$.[2]

The following result follows from [27, Definition 1.2.1 and Theorems 1.2.6, 1.2.7].

Theorem 4.1.7 *Suppose that a Bayesian network as in Definition 4.1.6 is given with a DAG* $\mathcal{G}$ *with vertex set* $V = \{X_1, \ldots, X_n\}$. *Then the following hold:*

(i) For every $X_j \in V$, X_j *and the set of all predecessors of* X_j *are independent over* $Par(X_j)$.

(ii) For every $X_j \in V$, X_j *and the set of all nondescendants of* X_j *(except* X_j *itself) are independent over* $Par(X_j)$.

[1]By 'minimal' I mean that no proper subset of $Par(X_j)$ has the given property.

[2]In other words, for all $i_1, \ldots, i_n \in \{0,1\}$, the probability $\mu(X_1 = i_1, \ldots, X_n = i_n)$ can be computed by using only the conditional probabilities associated with vertices of $\mathcal{G}$.

Moreover, if condition (i) or (ii) holds, then V can be ordered so that conditions (a) and (b) in Definition 4.1.6 hold without changing the arrows of the DAG $\mathcal{G}$.

Remark. We will use different names for probabilty distributions (and probability measures), so they will not always be called μ. Other common names in this text will be w or $\mathbb{P}$, possibly with some subscript.

4.2 The possible worlds approach

In the possible worlds approach to logic and probability we assume that we have a set of possible worlds, which we often denote $\mathbf{W}$, and a probability distribution $\mu : \mathbf{W} \to \mathbb{R}$. The idea is of course that each world (or "state") has certain probability which is given by μ.

If we work with propositional logic with propositional variables $p_1, \ldots, p_m$, then a possible world is (represented by) a truth assignment to $p_1, \ldots, p_m$, where a truth assignment is represented by a vector $\vec{a} \in \{0,1\}^m$. Thus $\mathbf{W}$ is a subset of $\{0,1\}^m$. From a probability distribution $\mu : \mathbf{W} \to \mathbb{R}$ we can derive a notion of probability of a formula $\varphi \in \mathsf{Prop}(p_1, \ldots, p_m)$. The general definitions, in the case of propositional logic, are given in Section 4.2.10. but we begin by illustrating the idea by a few examples in sections 4.2.1–4.2.4. The example in Section 4.2.4 gives a hint to the limitations of propositional logic and motivates using first-order logic in such situations. It is possible to read the general definitions in Section 4.2.10 first and then read the examples in sections 4.2.1–4.2.4.

Starting from Section 4.2.6 we use first-order logic as our modelling language, that is the language that we use to describe the world. When we work with first-order logic we need to fix a first-order language L (a set of relation, function and constant symbols) which is appropriate for "talking about" the relations, functions and objects that we are interested in. A possible world will in this context be an L-structure, so the set $\mathbf{W}$ of possible worlds is a set of L-structures. From a probability distribution $\mu : \mathbf{W} \to \mathbb{R}$ we can derive a notion probability of a first-order sentence (i.e. formula without free variables). The general definitions (in the context of first-order logic) are given in Section 4.2.10 and it is possible to read these first. Examples of using first-order logic and the possible worlds approach are given in Sections 4.2.6 – 4.2.9.

Note: *The examples are important because they illustrate how one can reason to determine the probability of complex formulas and because they illustrate how to derive a probability distribution on possible worlds from a Bayesian network.*

4.2.1 Example: propositional logic and the uniform probability distribution on truth assignments

Let p, q and r be propositional variables, representing propositions/assertions that may be true or false. Let $\mathsf{Prop}(p, q, r)$ be the set of all propositional formulas that can be constructed from p, q and r. The truth values 'true' and 'false' are encoded by the numbers 1 and 0, respectively. We represent a truth assignment to p, q and r by a binary vector, for example $(1, 0, 1)$ which assigns p the value 1 (true), q the value 0 (false) and r the value 1 (true). Each truth assignment to p, q and r corresponds to a possible world. For instance, the truth assignment $(1, 0, 1)$ represents a world in which p and r are true and q is false.

Note that $\{0,1\}^3$ denotes the set of binary vectors of length 3. A probability distribution on $\{0,1\}^3$ corresponds to a probability distribution on the set of possible worlds. Which probability distribution we choose depends on the application. Suppose that (for whatever reason) we think that the uniform probability distribution on $\{0,1\}^3$ is the appropriate.

Recall that the uniform probability distribution gives every $\vec{a} \in \{0,1\}^3$ the same probability, namely $\frac{1}{8}$ since there are $2^3 = 8$ truth assignments to p, q and r. Thus we define a probability distribution $\mu : \{0,1\}^3 \to [0,1]$ by

$$\mu(\vec{a}) \;=\; \frac{1}{8} \quad \text{for all } \vec{a} \in \{0,1\}^3 \quad \text{and we extend it to subsets of } \{0,1\}^3 \text{ by,}$$
$$\text{for every } A \subseteq \{0,1\}^3,\ \mu(A) = \sum_{\vec{a} \in A} \mu(\vec{a}).$$

So for example, $\mu(\{(1,0,1),(1,1,0),(0,1,0)\}) = \mu((1,0,1)) + \mu((1,1,0)) + \mu((0,1,0)) = 1/8 + 1/8 + 1/8 = 3/8$.

Let $\varphi \in \mathsf{Prop}(p,q,r)$. We will interpret the notion of "the probability that φ is true" as the probability, with respect to μ, of the event that a world (that is, truth assignment) satisfies φ (i.e. makes φ true). More precisely we define the probability of φ, denoted $\mathbb{P}_\mu(\varphi)$ as follows:

$$\text{For every } \varphi \in \mathsf{Prop}(p,q,r), \text{ let } \mathbb{P}_\mu(\varphi) := \mu(\{\vec{a} \in \{0,1\}^3 : \vec{a} \models \varphi\}).$$

Observe that if $X_\varphi : \{0,1\}^3 \to \{0,1\}$ is the binary random variable defined by $X_\varphi(\vec{a}) = 1$ if $\vec{a} \models \varphi$ and $X_\varphi(\vec{a}) = 0$ otherwise, then $\mathbb{P}_\mu(\varphi) = \mu(X_\varphi = 1)$ (where we use notation from Section 4.1).

Let us compute $\mathbb{P}_\mu(\varphi)$ for a concrete formula φ. Let φ denote the formula

$$((p \wedge q) \to r) \wedge \neg(q \wedge r \wedge \neg p).$$

By using a truth table (for example) we se that the following (and only the following) truth assignments make φ true: $(1,1,1),(1,0,1),(0,0,1),(0,1,0),(1,0,0),(0,0,0)$. It follows that

$$\mathbb{P}_\mu(\varphi) = \mu(\{(1,1,1),(1,0,1),(0,0,1),(0,1,0),(1,0,0),(0,0,0)\}) = 6 \cdot \frac{1}{8} = \frac{3}{4}.$$

We also see that

$$\mathbb{P}_\mu(p) = \mu(\{(1,1,1),(1,1,0),(1,0,1),(1,0,0)\} = 4 \cdot \frac{1}{8} = \frac{1}{2},$$

and in a similar way $\mathbb{P}_\mu(q) = \mathbb{P}_\mu(r) = 1/2$.

Moreover, with the given choice of μ, $\mathbb{P}_\mu(p)$ is independent from the truth values of q and r (and $\mathbb{P}_\mu(q)$ is independent of the truth value of r), more precisely,

$$\begin{aligned}
&\mathbb{P}_\mu(p \wedge q \wedge r) = \mathbb{P}_\mu(p) \cdot \mathbb{P}_\mu(q) \cdot \mathbb{P}_\mu(r),\\
&\mathbb{P}_\mu(p \wedge \neg q \wedge r) = \mathbb{P}_\mu(p) \cdot \mathbb{P}_\mu(\neg q) \cdot \mathbb{P}_\mu(r),\\
&\mathbb{P}_\mu(p \wedge q \wedge \neg r) = \mathbb{P}_\mu(p) \cdot \mathbb{P}_\mu(q) \cdot \mathbb{P}_\mu(\neg r), \quad \text{and}\\
&\mathbb{P}_\mu(p \wedge \neg q \wedge \neg r) = \mathbb{P}_\mu(p) \cdot \mathbb{P}_\mu(\neg q) \cdot \mathbb{P}_\mu(\neg r).
\end{aligned}$$

The tedious verification of this is left to the reader (but luckily all identities are proved in the same way by just modifying details).

4.2.2 Example: propositional logic with a probability distribution derived from a Bayesian network

As in the previous example, for each formula in $\mathsf{Prop}(p,q,r)$ we will associate a probability. But now we assume that the probabilities of p, q and r are given by the Bayesian network in Figure 4.1 where the (conditional) probabilities are denoted by w:

Figure 4.1:

$$\begin{tikzcd}
p \arrow[dr] & & q \arrow[dl] \\
& r &
\end{tikzcd}$$

$w(p) = 3/4$	$w(r \mid p \wedge q) = 3/4$
$w(q) = 1/3$	$w(r \mid \neg p \wedge q) = 1/2$
	$w(r \mid p \wedge \neg q) = 1/2$
	$w(r \mid \neg p \wedge \neg q) = 1/4$

The DAG in Figure 4.1 gives the information that p and q, viewed as binary random variables (i.e. taking values in $\{0,1\}$), are independent. Differently expressed, it tells that the probability that p is true is independent of the truth value of q and vice versa. The DAG also tells that r depends on p and q. The table for w tells that the probability that p is true is $3/4$ independently of the truth value of q and the the probability that q is true is $1/3$ independently of the truth value of p. The table also gives the conditional probabilities for r, given the truth values of p and q. So for example, the probability that r is true, given that both p and q are true, is $3/4$. Note that the event "both p and q are true" is encoded by the formula $p \wedge q$ and so on.

Recall that, just as in the previous example, the binary triple $(1,0,1)$ encodes the truth assignment, or "possible world", in which p is true, q is false and r is true. Given our Bayesian network we can compute the probability of the possible world $(1,0,1)$ as follows:

$$\begin{aligned}
&\text{probability of } (1,0,1) = \\
&(\text{probability that } r \text{ is true, given that } p \text{ is true and } q \text{ is false})\cdot \\
&(\text{probability that } p \text{ is true and } q \text{ is false}) = \\
&(\text{probability that } r \text{ is true, given that } p \text{ is true and } q \text{ is false})\cdot \\
&(\text{probability that } p \text{ is true}) \cdot (\text{probability that } q \text{ is false}) = \\
&w(r \mid p \wedge \neg q) \cdot w(p) \cdot w(\neg q) = \frac{1}{2} \cdot \frac{3}{4} \cdot \frac{2}{3} = \frac{1}{4}.
\end{aligned}$$

In the same way we can, for every $\vec{a} \in \{0,1\}^3$, compute the probability of the truth assignment (or possible world) $\vec{a}$. Denote this probability by $w(\vec{a})$. So for example, above we found that $w((1,0,1)) = 1/4$. We can extend w to subsets of $\{0,1\}^3$ as follows, analogously as for μ in Example 4.2.1:

$$\text{For every } A \subseteq \{0,1\}^3, \ w(A) := \sum_{\vec{a} \in A} w(\vec{a}).$$

As in Example 4.2.1 it makes sense to define the "probability of φ" (where $\varphi \in \mathsf{Prop}(p,q,r)$) in the following way, now using w since w is now our probability distribution on $\{0,1\}^3$:

$$\text{For every } \varphi \in \mathsf{Prop}(p,q,r), \text{ let } \mathbb{P}_w(\varphi) := w(\{\vec{a} \in \{0,1\}^3 : \vec{a} \models \varphi\}).$$

Consider again the formula $((p \wedge q) \to r) \wedge \neg(q \wedge r \wedge \neg p)$ from Example 4.2.1 and denote it by φ. As mentioned in that example, φ is satisfied by the following (and only the following) truth assignments: $(1,1,1), (1,0,1), (0,0,1), (0,1,0), (1,0,0), (0,0,0)$. Thus we get

$$\begin{aligned}&\mathbb{P}_w(\varphi) = w(\{(1,1,1),(1,0,1),(0,0,1),(0,1,0),(1,0,0),(0,0,0)\}) = \\ &w((1,1,1)) + w((1,0,1)) + w((0,0,1)) + w((0,1,0)) + w((1,0,0)) + w((0,0,0)) = \\ &3/16 + 1/4 + 1/24 + 1/24 + 1/4 + 1/8 = 43/48.\end{aligned}$$

In other words, in this setting, the probability that φ is true is 43/48.

We can also compute $\mathbb{P}_w(\varphi)$ in a different way, which in this case makes the computations easier. Consider the probability that $\neg\varphi$ is true, that is, $\mathbb{P}_w(\neg\varphi)$. A truth assignment satisfies $\neg\varphi$ if and only if it does not satisfy φ, so the following list contains all truth assignments which satisfy $\neg\varphi$: $(1,1,0)$, $(0,1,1)$. Hence $\mathbb{P}_w(\neg\varphi) = w(\{(1,1,0),(0,1,1)\}) = w((1,1,0)) + w((0,1,1)) = 1/16 + 1/24 = 5/48$. Since w is a probability distribution we have $\sum_{\vec{a}\in\{0,1\}^3} w(\vec{a}) = 1$ and therefore $\mathbb{P}_w(\varphi) = 1 - \mathbb{P}_w(\neg\varphi) = 1 - 5/48 = 43/48$.

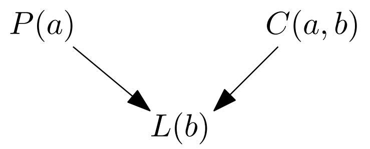
4.2.3 Example: using a hybrid of propositional and first-order logic to model a situation

In this example it is natural to use first-order logic to express certain statements, but since the statements only talk about a fixed and small set of objects it is convenient to use, essentially, the semantics from propositional logic when dealing with truth or falsity of formulas.

Suppose that we want to describe the following situation. We have a power station, called a, and a lighthouse, called b. We also have an electrical cable connecting a to b. With probability 0.8 the power station delivers electricity. Independently of whether the power station delivers power or not, the probability that the electrical cable is working is 0.7. Conditioned on the event that the power station delivers electricity and that the electrical cable is working, the probability that the light in the lighthouse is on is 0.9. (Obviously the infrastructure here is rather poor.)

Let $P(a)$ stand for the proposition "the power station a delivers electricity", let $C(a,b)$ stand for the proposition "the electrical connection (i.e. cable) between a and b is working" and let $L(b)$ stand for the proposition "the light in the lighthouse b is on". The stated probabilities are described by the Bayesian network in Figure 4.2.

Figure 4.2:


$$\begin{tikzcd}P(a) \arrow[dr] & & C(a,b) \arrow[dl] \\ & L(b) & \end{tikzcd}$$

$w(P(a)) = 0.8$	$w(L(b) \mid P(a) \wedge C(a,b)) = 0.9$
$w(C(a,b)) = 0.7$	$w(L(b) \mid \neg P(a) \wedge C(a,b)) = 0$
	$w(r \mid P(a) \wedge \neg C(a,b)) = 0$
	$w(r \mid \neg P(a) \wedge \neg C(a,b)) = 0$

Note that if we rename the propositions with $p := P(a)$, $q := C(a,b)$ and $r := L(b)$, then we get the same directed acyclic graph as in the example in Section 4.2.2, but the probabilities are different. Also observe that we can simplify the table of probabilities by replacing the last three conditional probabilities by $w(L(b) \mid \neg(P(a) \wedge C(a,b))) = 0$.

In the same way as in the example in Section 4.2.2 we can compute the probability $w(\vec{a})$ of any truth assignment (or "possible world") $\vec{a} \in \{0.1\}^3$ to $P(a)$, $C(a,b)$ and $L(b)$, where, for example, $(1,0,1)$ signifies that $P(a)$ is true, $C(a,b)$ is false and $L(b)$ is true. From the Bayesian network we have, for example, that $w((1,0,1)) = 0.8 \cdot 0.3 \cdot 0 = 0$ and $w((1,1,1)) = 0.8 \cdot 0.7 \cdot 0.9 = 0.504$. We can define $\mathbb{P}_w(\varphi)$ for every $\varphi \in \mathsf{Prop}(P(a), C(a,b), L(b))$ in the same way as in Example 4.2.2. So we have, for example:

$$\begin{aligned}
\mathbb{P}_w(L(b)) =& w(\{(1,1,1),(1,0,1),(0,1,1),(0,0,1)\}) = \\
& w((1,1,1)) + w((1,0,1)) + w((0,1,1)) + w((0,0,1)) = \\
& 0.504 + 0 + 0 + 0 = 0.504, \\
\mathbb{P}_w(\neg L(b)) =& 1 - 0.504 = 0.496, \\
\mathbb{P}_w(P(a) \wedge \neg L(b)) =& w(\{(1,0,0),(1,1,0)\}) = w((1,0,0)) + w((1,1,0)) = \\
& 0.8 \cdot 0.3 \cdot 1 + 0.8 \cdot 0.7 \cdot 0.1 = 0.296.
\end{aligned}$$

We can also compute (for example) the conditional probability that $C(a,b)$ is true if we know that $P(a) \wedge \neg L(b)$ is true:

$$\mathbb{P}_w(C(a,b) \mid P(a) \wedge \neg L(b)) = \frac{\mathbb{P}_w(C(a,b) \wedge P(a) \wedge \neg L(b))}{\mathbb{P}_w(P(a) \wedge \neg L(b))} = \frac{0.056}{0.296} \approx 0.189.$$

4.2.4 Example: towards abandoning propositional logic for modelling purposes

With this example I want to illustrate that if we want to model a situation with many (perhaps unknown) objects (perhaps of an unknown number), then it may be more convenient to do it by using first-order logic and its semantics rather than propositional logic.

In this example we start from the same idea as in the example in Section 4.2.3, but now we consider 3 power stations, denoted a_1, a_2 and a_3 and, for each $i = 1,2,3$, an electric connection between a_i and the light house, denoted b. For each $i = 1,2,3$, $P(a_i)$ stands for the proposition "the power station a_i delivers electricity", $C(a_i, b)$ stands for the proposition "the electrical connection between a_i and b is working" and $L(b)$ stands for the proposition "the light in the lighthouse b is on". The probabilities and dependencies are essentially the same as in the example in Section 4.2.3 as depicted by the Bayesian network in Figure 4.3 where the notation $\bigvee_{i=1}^{3}(P(a_i) \wedge C(a_i,b))$ is an abbreviation for

$$(P(a_1) \wedge C(a_1,b)) \vee (P(a_2) \wedge C(a_2,b)) \vee (P(a_3) \wedge C(a_3,b)).$$

The Bayesian network tells that the events $P(a_1), P(a_2), P(a_3), C(a_1,b), C(a_2,b)$ and $C(a_3,b)$ are independent of each other. For each $i = 1,2,3$, $P(a_i)$ has probability 0.8 and $C(a_i,b)$ has probability 0.7. We also read from the Bayesian network that, conditioned on the event that $P(a_i) \wedge C(a_i,b)$ is true for at least one i, the probability that $L(b)$ is true is 0.9. Conditioned on the complementary event that $P(a_i) \wedge C(a_i,b)$ is false for every i, the probability that $L(b)$ is true is 0.

In this example we have 7 propositional variables:

$$P(a_1), P(a_2), P(a_3), C(a_1,b), C(a_2,b), C(a_3,b), L(b).$$

Therefore a truth assignment, or "possible world", is encoded by a 7-tuple $\vec{a} \in \{0,1\}^7$. We adopt the convention that the digits in a binary 7-tuple $\vec{a}$ refer to the propositional variables in the order listed above. So if $\vec{a} = (1,0,\dots)$ then $\vec{a}$ assigns $P(a_1)$ the truth value "true" and $P(a_2)$ the truth value "false".

Figure 4.3:

$$\begin{tikzcd}
P(a_1) \arrow[drr] & P(a_2) \arrow[dr] & P(a_3) \arrow[d] & C(a_1,b) \arrow[dl] & C(a_2,b) \arrow[dll] & C(a_3,b) \arrow[dlll] \\
 & & L(b) & & &
\end{tikzcd}$$

For $i = 1, 2, 3$, $w(P(a_i)) = 0.8$	$w\big(L(b) \mid \bigvee_{i=1}^{3}(P(a_i) \wedge C(a_i, b))\big) = 0.9$
For $i = 1, 2, 3$, $w(C(a_i, b)) = 0.7$	$w\big(L(b) \mid \neg \bigvee_{i=1}^{3}(P(a_i) \wedge C(a_i, b))\big) = 0$

Just as in previous examples, if

$$\varphi \in \mathsf{Prop}(P(a_1), P(a_2), P(a_3), C(a_1, b), C(a_2, b), C(a_3, b), L(b))$$

then we let $\mathbb{P}_w(\varphi) := w(\{\vec{a} \in \{0,1\}^7 : \vec{a} \models \varphi\})$, but now the probability distribution w on $\{0,1\}^7$ is determined by the Bayesian network of this example. Since there are $2^7 = 128$ truth assignments calculations by hand are not in general as easy as in Example 4.2.3. But the independencies described by the Bayesian network allow us to simplify some calculations. For example, using the independencies described by the Bayesian network we have, for every $i = 1, 2, 3$,

$$\mathbb{P}_w(P(a_i) \wedge C(a_i, b)) = 0.8 \cdot 0.7 = 0.56,$$

so

$$\mathbb{P}_w(\neg(P(a_i) \wedge C(a_i, b))) = 1 - 0.56 = 0.44.$$

By the independencies again and some basic combinatorics,

$$\mathbb{P}_w\Big(\bigvee_{i=1}^{3}(P(a_i) \wedge C(a_i, b))\Big) = (0.56)^3 + 3 \cdot 0.56 \cdot (0.44)^2 + 3 \cdot (0.56)^2 \cdot 0.44 = 0.914816.$$

Since $\mathbb{P}_w\Big(L(b) \mid \neg \bigvee_{i=1}^{3}(P(a_i) \wedge C(a_i, b))\Big) = 0$ we get

$$\begin{aligned}
\mathbb{P}_w(L(b)) =& \mathbb{P}_w\Big(L(b) \mid \bigvee_{i=1}^{3}(P(a_i) \wedge C(a_i, b))\Big) \cdot \mathbb{P}_w\Big(\bigvee_{i=1}^{3}(P(a_i) \wedge C(a_i, b))\Big) \\
=& 0.9 \cdot 0.914816 \approx 0.823.
\end{aligned}$$

Now suppose that, for some large number n, we have n power stations $a_1, \ldots, a_n$ where each a_i has an electrical connection to b. We can add new propositional variables, denoted $P(a_4), P(a_5), \ldots, P(a_n), C(a_4, b), C(a_5, b), \ldots, C(a_n, b)$ where, for each $i = 1, \ldots, n$, $P(a_i)$ and $C(a_i, b)$ have the same meaning as before. Moreover assume that the probabilities are still as in the table of Figure 4.3, but with 3 replaced by n. So for example, for every $i = 1, \ldots, n$, $w(P(a_i)) = 0.8$.

We can also extend the directed acyclic graph of the Bayesian network with

$$P(a_4), P(a_5), \ldots, C(a_4, b), C(a_5, b), \ldots$$

in the obvious way; that is, for each on of these new random variables we add an arrow from it to $L(b)$. Now we can reason about probabilities as in the case when $n = 3$. But if n is large we also get a large Bayesian network. Since we have certain uniformities in the network (such as all $P(a_i)$ having the same probability) we can represent the same (in)dependencies and probabilities more succinctly by the so called *lifted Bayesian network* in figure 4.4:

Figure 4.4:

$$\begin{tikzcd} P(x) \arrow[dr] & & C(x,b) \arrow[dl] \\ & L(b) & \end{tikzcd}$$

$w(P(x)) = 0.8$	$w\big(L(b) \mid \exists x(P(x) \wedge C(x,b))\big) = 0.9$
$w(C(x,b)) = 0.7$	$w\big(L(b) \mid \neg\exists x(P(x) \wedge C(x,b))\big) = 0$

The lifted Bayesian network should be interpreted in the following way. Given that $P(x) \wedge C(x,b)$ holds for at least one power station x, the probability that $L(b)$ holds is 0.9; otherwise the probability of $L(b)$ is 0. For every power station x, the probability that $P(x)$ holds is 0.8 and this event is independent of the events $C(x,b)$, $P(y)$ and $C(y,b)$ for $y \neq x$. For every power station x, the probability that $C(x,b)$ holds is 0.7 and this event is independent of the events $P(x)$, $P(y)$ and $C(y,b)$ for $y \neq x$.

4.2.5 General definitions of probabilities in the the "possible worlds" approach for propositional logic

In each of Sections 4.2.1–4.2.4 we have seen a definition of probability distribution on a set of possible worlds (or possible states), where the set of possible worlds has been a set of truth assignments, and in each case a notion of probability of a formula (in propositional logic) was defined. Each one of those definitions is an instance of the general definition of a probability distribution on a set of possible worlds given in this section and the subsequent notion of probability of a formula (in propositional logic).

Let $\mathbf{P}$ be a finite or countable set of propositional variables (so each $p \in \mathbf{P}$ represents a statement which can be true or false). Let $\mathbf{W}$ be a set of truth assignments to members of $\mathbf{P}$ and let $\mu : \mathbf{W} \to \mathbb{R}$ be a probability distribution. Since $\mathbf{P}$ is countable, perhaps even finite, we can fix an ordering of the members of $\mathbf{P}$: $\mathbf{P} = \{p_0, p_1, p_2, \ldots\}$. Therefore we think of a truth assignment as a sequence $\vec{a} = (b_0, b_1, b_2, \ldots)$ where each b_i is 0 or 1 and b_i is the truth value of p_i that is assigned by $\vec{a}$.

The probability distribution $\mu : \mathbf{W} \to \mathbb{R}$ can be extended to a *probability measure* on the set of all subsets of $\mathbf{W}$, which we give the same name μ, as follows:

$$\text{For every } \mathbf{X} \subseteq \mathbf{W}, \mu(\mathbf{X}) := \sum_{\vec{a} \in \mathbf{X}} \mu(\vec{a}).$$

For all $\varphi \in \mathsf{Prop}(\mathbf{P})$ we define

$$\mathbb{P}_\mu(\varphi) := \mu\big(\{\vec{a} \in \mathbf{W} : \vec{a} \models \varphi\}\big)$$

and for all $\varphi, \psi \in \mathsf{Prop}(\mathbf{P})$ we define

$$\mathbb{P}_\mu(\varphi \mid \psi) := \frac{\mathbb{P}_\mu(\varphi \wedge \psi)}{\mathbb{P}_\mu(\psi)}.$$

We read '$\mathbb{P}_\mu(\varphi)$' as "the probability of φ" or "the probability that φ is true", or something similar. The notation "$\mathbb{P}_\mu(\varphi \mid \psi)$" is read as "the (conditional) probability that φ is true given that ψ is true", or something similar. The following follows in a straightforward way from the definition:

Proposition 4.2.1 (Basic properties of $\mathbb{P}_\mu$)

(a) *If φ and ψ are logically equivalent formulas, then $\mathbb{P}_\mu(\varphi) = \mathbb{P}_\mu(\psi)$.*

(b) *$\mathbb{P}_\mu(\neg\varphi) = 1 - \mathbb{P}_\mu(\varphi)$ for all formulas φ.*

(c) *Suppose that every truth assignment in $\mathbf{W}$ makes $\varphi \wedge \psi$ false, in other words that there is no "world" in which both φ and ψ are true. Then $\mathbb{P}_\mu(\varphi \vee \psi) = \mathbb{P}_\mu(\varphi) + \mathbb{P}_\mu(\psi)$.*

(d) *If φ is a tautology, also called valid formula, (that is, a formula which is true for every truth assignment), then $\mathbb{P}_\mu(\varphi) = 1$.*

Remark 4.2.2 (Viewing formulas as binary random variables) Suppose that $\mathbf{P}$ is a set of propositional variables and $\mathbf{W}$ a set of truth assignments to the members of $\mathbf{P}$. To every $\varphi \in \mathsf{Prop}(\mathbf{P})$ we can associate a binary random variable $X_\varphi : \mathbf{W} \to \{0,1\}$ defined by $X_\varphi(\vec{a}) = 1$ if $\vec{a} \models \varphi$ and $X_\varphi(\vec{a}) = 0$ otherwise. If $\mu : \mathbf{W} \to \mathbb{R}$ is a probability distribution, then, for every $\varphi \in \mathsf{Prop}(\mathbf{P})$, we have

$$\mathbb{P}_\mu(\varphi) = \mu\big(\{\vec{a} \in \mathbf{W} : \vec{a} \models \varphi\}\big) = \mu\big(\{\vec{a} \in \mathbf{W} : X_\varphi(\vec{a}) = 1\}\big) = \mu\big(X_\varphi = 1\big).$$

Thus, $\mathbb{P}_\mu(\varphi)$ is equal to the probability (using μ) that the random variable X_φ has the value 1.

Remark 4.2.3 (The probability of logical implication is *not* the same as conditional probability) Suppose that $\mathbf{W}$ is a set of truth assignments and that $\mu : \mathbf{W} \to \mathbb{R}$ is a probability distribution. A propositional formula of the form $\psi \to \varphi$ expresses that "if ψ is true then φ is true". $\mathbb{P}_\mu(\varphi \mid \psi)$ is the probability that φ is true *if* we already know that ψ is true. One may be tempted to believe that $\mathbb{P}_\mu(\psi \to \varphi) = \mathbb{P}_\mu(\varphi \mid \psi)$, but this is *not* true in general. This will be illustrated by an example, but first a conceptual explanation will be given. The formula $\psi \to \varphi$ is logically equivalent to $\neg\psi \vee \varphi$ so if ψ is made false by most truth assignments in $\mathbf{W}$ then $\mathbb{P}_\mu(\psi \to \varphi)$ is high, but it is consistent with this situation that $\vec{a} \not\models \varphi$ for most (or even all) $\vec{a} \in \mathbf{W}$ such that $\vec{a} \models \psi$ and then $\mathbb{P}_\mu(\varphi \mid \psi)$ is low.

Now we illustrate this by a concrete example, namely the example in Section 4.2.1. In that example p, q and r denote propositional variables and μ is the uniform probability distribution on $\mathbf{W}$ which is the set of all 8 truth assignments to p, q and r. Since $(p \wedge q) \to r$ is true in 7 out of 8 truth assignments it follows that $\mathbb{P}_\mu\big((p \wedge q) \to r\big) = 7/8$. (Note that $p \wedge q$ is false in 6 truth assignments, a rather high proportion.) On the other hand we have

$$\mathbb{P}_\mu(r \mid p \wedge q) = \frac{\mathbb{P}_\mu(r \wedge p \wedge q)}{\mathbb{P}_\mu(p \wedge q)} = \frac{1/8}{2/8} = 1/2,$$

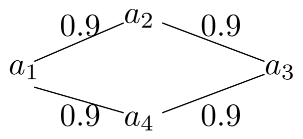
so clearly $\mathbb{P}_\mu\big((p \wedge q) \to r\big) \neq \mathbb{P}_\mu(r \mid p \wedge q)$.

There is nothing particular with the example in Section 4.2.1. For each one of the examples given in Sections 4.2.2–4.2.4 one can find formulas φ and ψ such that the probability of $\psi \to \varphi$ is different from the conditional probability of φ given ψ.

4.2.6 Example: a network and its possible states modelled by first-order logic

In contrast to the previous examples we will from now on use first-order logic as our modelling language (i.e. language that we describe the world with). Thus, from now on a possible world will be a first-order structure for some language choosen to fit the application. Suppose that we have a little network with 4 vertices a_1, a_2, a_3 and a_4 and "cables" between some of them according to the graph in Figure 4.5.

Figure 4.5:



We assume that each one of the cables, indicated by an edge, has a probability of 0.9 of being open, independently of the status of the other cables. We would like to formalize this context by first-order logic. In other words, the "possible worlds" should be first-order structures for an appropriate first-order language. In this example the objects are a_1, a_2, a_3, a_4 so we consider first-order structures with domain $D = \{a_1, a_2, a_3, a_4\}$. Let O be a binary relation symbol. For $a, b \in D$ we think of $O(\hat{a}, \hat{b})$ as asserting that "the connection between a and b is open", where $\hat{a}$ and $\hat{b}$ are constant symbols which are interpreted as a and b, respectively. This choice of language seems appropriate because the only relationship of relevance in this example is the "connectedness relation". The first-order language that we work with will have O as its only relation symbol (and it has no function symbols and no constant symbols besides $\hat{a}$ for every $a \in D$).

Now let $\mathbf{W}$ (the set of possible worlds) be the set of all first-order structures $\mathcal{A}$ for this language which have domain D and such that, for every $a \in D$, $\hat{a}$ is interpreted as a and O is interpreted as a *symmetric relation*, that is, for all $a, b \in D$, if $\mathcal{A} \models O(\hat{a}, \hat{b})$ then $\mathcal{A} \models O(\hat{b}, \hat{a})$.[3] Note that $\mathbf{W}$ contains structures $\mathcal{A}$ in which (for example) a connection between a_2 and a_4 exists and is open, because nothing prevents O from being interpreted in such a way that $(a_2, a_4) \in O^{\mathcal{A}}$ in which case $\mathcal{A} \models O(\hat{a}_2, \hat{a}_4)$. This situation is not in accordance with our initial assumptions because there is no cable between a_2 and a_4 and therefore there cannot be an open cable between a_2 and a_4. However we will make sure that every structure that violates any of our initial assumptions will have probability 0. (Alternatively we could have included in $\mathbf{W}$ only the structures that do not violate our assumptions which are illustrated by Figure 4.5.)

We now define a probability distribution μ on $\mathbf{W}$ which will formalize the assumptions we made in the beginning. For example, according to the assumptions, the probability that the cables from a_1 to a_2 and from a_4 to a_3 are open and the other cables closed is $(0.9)^2(0.1)^2$. The probability distribution μ will reflect this. For every $\mathcal{A} \in \mathbf{W}$,

if $\mathcal{A} \models O(\hat{a}_1, \hat{a}_3)$ or $\mathcal{A} \models O(\hat{a}_2, \hat{a}_4)$, then $\mu(\mathcal{A}) = 0$ (these are the $\mathcal{A}$'s which violate our assumptions),

otherwise, $\mu(\mathcal{A}) = (0.9)^n(0.1)^{4-n}$ where n is the number of pairs $(a_i, a_j) \in D^2$ such that $i < j$ and $\mathcal{A} \models O(\hat{a}_i, \hat{a}_j)$. (The constraint $i < j$ is there to prevent us from

[3] This is a natural assumption if we have the idea that something can simultaneously flow through the cables in both directions. But one can make other choices depending on the application.

counting a cable twice as both pairs (a_i, a_j) and (a_j, a_i) represent the same cable.)

We extend μ to subsets of $\mathbf{W}$ in the following way:

$$\text{For every } X \subseteq \mathbf{W}, \ \ \mu(X) = \sum_{\mathcal{A} \in X} \mu(\mathcal{A}).$$

For every sentence φ (in the language that we consider) we define "the probability that φ is true", denoted $\mathbb{P}_\mu(\varphi)$ (where the index μ indicates that the probabilities on sentences depend on μ), as follows:

$$\mathbb{P}_\mu(\varphi) = \mu\big(\{\mathcal{A} \in \mathbf{W} : \mathcal{A} \models \varphi\}\big).$$

As an example we will compute $\mathbb{P}_\mu(\varphi)$ in the case when φ is $\exists x(O(\hat{a}_1, x) \wedge O(x, \hat{a}_3))$ which (in the given context) expresses that "there is an open path of cables all the way from a_1 to a_3". Suppose that $\mathcal{A} \in \mathbf{W}$ and $\mu(\mathcal{A}) > 0$. We consider the cases in which $\mathcal{A} \models \varphi$.

- If all cables are open in $\mathcal{A}$ (that is, if $\mathcal{A} \models O(\hat{a}_1, \hat{a}_2) \wedge O(\hat{a}_2, \hat{a}_3) \wedge O(\hat{a}_1, \hat{a}_4) \wedge O(\hat{a}_4, \hat{a}_3)$, then $\mu(\mathcal{A}) = (0.9)^4$.

- If exactly three of the four cables are open in $\mathcal{A}$ then $\mu(\mathcal{A}) = (0.9)^3(0.1)$. There are four such possibilities (i.e. four different structures satisfying this condition).

- If $\mathcal{A} \models O(\hat{a}_1, \hat{a}_i) \wedge O(\hat{a}_i, \hat{a}_3) \wedge \neg O(\hat{a}_1, \hat{a}_j) \wedge \neg O(\hat{a}_j, \hat{a}_3)$ where either $i = 2$ and $j = 4$ or $i = 4$ and $j = 2$, then $\mu(\mathcal{A}) = (0.9)^2(0.1)^2$. There are two such possibilities.

It follows that $\mathbb{P}_\mu(\varphi) =$

$$\mu\big(\{\mathcal{A} \in \mathbf{W} : \mathcal{A} \models \varphi\}\big) = (0.9)^4 + 4 \cdot (0.9)^3 \cdot (0.1) + 2 \cdot (0.9)^2 \cdot (0.1)^2 = 0.9963.$$

Remark 4.2.4 We could also have formalized the context of this example with propositional logic. We could have viewed the formulas $O(\hat{a}_1, \hat{a}_2)$, $O(\hat{a}_1, \hat{a}_4)$, $O(\hat{a}_2, \hat{a}_3)$ and $O(\hat{a}_3, \hat{a}_4)$ as propositional variables (statements that can be true or false). In the structures in $\mathbf{W}$ the formula $\exists x(O(\hat{a}_1, x) \wedge O(x, \hat{a}_3))$ holds if and only if the formula

$$\big(O(\hat{a}_1, \hat{a}_2) \wedge O(\hat{a}_2, \hat{a}_3)\big) \vee \big(O(\hat{a}_1, \hat{a}_4) \wedge O(\hat{a}_4, \hat{a}_3)\big)$$

holds. If we have fixed a finite domain and if each element in the domain is "named" by some constant symbol, then every first-order sentence will be equivalent to a sentence without quantifiers where a sentence without quantifiers is essentially a formula in propositional logic. However, if we formalize a context with large domain in propositional logic, then we may need very many propositional variables while if we formalize the same context in first-order logic the number of relation symbols needed is independent of the size of the domain. More precisely, when formalizing a context in first-order logic the (first-order) language that we choose is independent of the size of the domain. It only depends on what kinds of "relationships" we want to formalize. Furthermore, any first-order sentence describes the same property (that a structure, or "possible world", may have) independently of the size of the domain. On the other hand, if we want to translate a first-order sentence to propositional logic, as we did above, the translation will often depend on the size of the domain. Thus it is fair to say formalizations in first-order logic are "stable" under changes of the domain.

4.2.7 Example: many power stations and a lighthouse modelled in first-order logic

In this example we consider a similar situation as in the examples in Sections 4.2.4 and 4.2.4. Suppose that we have a light house b and 100 power stations, for simplicity called $1, 2, \ldots, 100$. From each power station i there is an electric connection to b. (This may be a bit unrealistic but the point is to show that when we consider many objects it may be more natural to use a formalization in first-order logic instead, as argued in the end of Section 4.2.6.) We assume the following: the probability that power station i delivers electricity is 0.8, the probability that the connection between the power station i and the light house b is working is 0.7. All these events are assumed to be independent of each other. Conditioned on the event that there is some power station i such that i delivers electricity and the electric connection between i and b is working, the probability is 0.9 that the light is on in the light house b. We use relations symbols P, C and L, where P and L are unary and C is binary, to formalize the situation. The probabilities are described by the *lifted Bayesian network* in figure 4.4.

As the set of possible worlds $\mathbf{W}$ we take the set of structures $\mathcal{A}$ (for the described first-order language) with domain $D = \{1, \ldots, 100, b\}$ such that

- $\mathcal{A} \models \neg L(\hat{a})$ for all $a \in \{1, \ldots, 100\}$,
- $\mathcal{A} \models \neg P(\hat{b})$, and
- $\mathcal{A} \models \neg C(\hat{\imath}, \hat{\jmath})$ for all $i, j \in \{1, \ldots, 100\}$.

We now define a probability distribution w on $\mathbf{W}$ which captures the assumptions that we have made. As a preliminary step we first define a function w_0 on $\mathbf{W}$ as follows: For every $\mathcal{A} \in \mathbf{W}$, if α is the number of $i \in \{1, \ldots, 100\}$ such that $\mathcal{A} \models P(\hat{\imath})$ and β is the number of $i \in \{1, \ldots, 100\}$ such that $\mathcal{A} \models C(\hat{\imath}, \hat{b})$, then

$$w_0(\mathcal{A}) = (0.8)^{\alpha}(0.2)^{100-\alpha}(0.7)^{\beta}(0.3)^{100-\beta}.$$

For every $\mathcal{A} \in \mathbf{W}$,

1. if $\mathcal{A} \models \exists x(P(x) \wedge C(x, \hat{b})) \wedge L(\hat{b})$, then $w(\mathcal{A}) = w_0(\mathcal{A}) \cdot 0.9$,
2. if $\mathcal{A} \models \exists x(P(x) \wedge C(x, \hat{b})) \wedge \neg L(\hat{b})$, then $w(\mathcal{A}) = w_0(\mathcal{A}) \cdot 0.1$,
3. if $\mathcal{A} \models \neg\exists x(P(x) \wedge C(x, \hat{b})) \wedge L(\hat{b})$, then $w(\mathcal{A}) = w_0(\mathcal{A}) \cdot 0 = 0$, and
4. if $\mathcal{A} \models \neg\exists x(P(x) \wedge C(x, \hat{b})) \wedge \neg L(\hat{b})$, then $w(\mathcal{A}) = w_0(\mathcal{A}) \cdot 1 = w_0(\mathcal{A})$.

We can extend w to a function on all subsets of $\mathbf{W}$ as follows:

$$\text{For every } X \subseteq \mathbf{W}, \ \ w(X) = \sum_{\mathcal{A} \in X} w(\mathcal{A}).$$

For every sentence φ (in the first-order language that we consider) we can define "the probability that φ is true (in a possible world)", denoted $\mathbb{P}_w(\varphi)$, in the usual way:

$$\text{For every sentence } \varphi, \ \mathbb{P}_w(\varphi) = w\big(\{\mathcal{A} \in \mathbf{W} : \mathcal{A} \models \varphi\}\big).$$

Due to the independencies in this example we have $\mathbb{P}_w(P(\hat{i}) \wedge C(\hat{i},\hat{b})) = 0.8 \cdot 0.7 = 0.56$ for all $i \in \{1,\dots,100\}$. Hence $\mathbb{P}_w\big(\neg(P(\hat{i}) \wedge C(\hat{i},\hat{b}))\big) = 1 - 0.56 = 0.44$. By the independencies again, $\mathbb{P}_w\big(\forall x \neg(P(x) \wedge C(x,\hat{b}))\big) = (0.44)^{100}$. Since $\forall x \neg(P(x) \wedge C(x,\hat{b}))$ is logically equivalent to $\neg\exists x(P(x) \wedge C(x,\hat{b}))$ it follows that $\mathbb{P}_w\big(\exists x(P(x) \wedge C(x,\hat{b}))\big) = 1 - (0.44)^{100}$. From 1 above it follows that

$$\mathbb{P}_w\big(L(\hat{b}) \wedge \exists x(P(x) \wedge C(x,\hat{b}))\big) = \big(1 - (0.44)^{100}\big) \cdot 0.9$$

which gives

$$\mathbb{P}_w\big(L(\hat{b}) \mid \exists x(P(x) \wedge C(x,\hat{b}))\big) = \frac{\mathbb{P}_w\big(L(\hat{b}) \wedge \exists x(P(x) \wedge C(x,\hat{b}))\big)}{\mathbb{P}_w\big(\exists x(P(x) \wedge C(x,\hat{b}))\big)} = 0.9.$$

Hence

$$\begin{aligned}\mathbb{P}_w\big(L(\hat{b})\big) &= \mathbb{P}_w\big(L(\hat{b}) \mid \exists x(P(x) \wedge C(x,\hat{b}))\big) \cdot \mathbb{P}_w\big(\exists x(P(x) \wedge C(x,\hat{b}))\big) \\ &= 0.9 \cdot \big(1 - (0.44)^{100}\big).\end{aligned}$$

Since $\big(1 - (0.44)^{100}\big)$ is extremely close to 1 it follows that $\mathbb{P}_w\big(L(\hat{b})\big)$ is extremely close to 0.9.

4.2.8 Example: where we consider a set of structures on a large finite domain with the uniform probability distribution

Let the domain be $D = \{1,\dots,n\}$, so the domain has size n. Let us consider a first-order language L with two relation symbols, Q and R, with arities 1 and 2, respectively, and with a constant symbol $\hat{k}$ for every $k \in D$. Let $\mathbf{W}$ be the set of all L-structures with domain D such that for every $k \in \{1,\dots,n\}$, $\hat{k}$ is interpreted as k. Intuitively speaking $\mathbf{W}$ is the set of possible worlds that we consider. Let μ be the uniform probability distribution on $\mathbf{W}$, in other words μ assigns the same probability to every structure (or possible world) $\mathcal{A} \in \mathbf{W}$. But what is $\mu(\mathcal{A})$?

There are 2^n possible interpretations of Q on D (since there are 2^n different subsets of D). There are 2^{n^2} possible interpretations of R on D (since $D^2 = D \times D$ has size n^2 and there are 2^{n^2} different subsets of D^2). As we can interpret Q and R in whichever way we like independently of each other there are exactly $2^n \cdot 2^{n^2} = 2^{n^2+n}$ structures in $\mathbf{W}$. Since μ gives each $\mathcal{A} \in \mathbf{W}$ the same probability we must have $\mu(\mathcal{A}) = 1/2^{n^2+n}$ for every $\mathcal{A} \in \mathbf{W}$.

We can extend μ to the set of all subsets of $\mathbf{W}$ as follows:

$$\text{For every } \mathbf{X} \subset \mathbf{W}, \;\; \mu(\mathbf{X}) = \sum_{\mathcal{A} \in \mathbf{X}} \mu(\mathcal{A}).$$

For every L-sentence φ we define "the probability that φ is true in a structure (or world) in $\mathbf{W}$", denoted $\mathbb{P}_\mu(\varphi)$ as follows:

$$\text{For every } L\text{-sentence } \varphi, \;\; \mathbb{P}_\mu(\varphi) = \mu\big(\{\mathcal{A} \in \mathbf{W} : \mathcal{A} \models \varphi\}\big).$$

As may be expected, for all $i, j \in D$, the probability of $Q(\hat{i})$ is 1/2 and the probability of $R(\hat{i},\hat{j})$ is 1/2. This follows from the following computations. There are exactly 2^{n^2+n-1} structures $\mathcal{A} \in \mathbf{W}$ such that $\mathcal{A} \models Q(\hat{i})$ (because now we have fixed the interpretation of Q on i). Since $\mu(\mathcal{A}) = 1/2^{n^2+n}$ for every $\mathcal{A} \in \mathbf{W}$ we get

$$\begin{aligned}&\mathbb{P}_\mu(Q(\hat{i})) = \mu\big(\{\mathcal{A} \in \mathbf{W} : \mathcal{A} \models Q(\hat{i})\}\big) = \\ &\sum_{\mathcal{A} \in \mathbf{W} \text{ and } \mathcal{A} \models Q(\hat{i})} \mu(\mathcal{A}) = 2^{n^2+n-1} \cdot \mu(\mathcal{A}) = 2^{n^2+n-1} \cdot \frac{1}{2^{n^2+n}} = 1/2.\end{aligned}$$

In a similar way (exercise) one can show that $\mathbb{P}_\mu(R(\hat{\imath},\hat{\jmath})) = 1/2$.

The following also holds:

(a) If $i, j \in D$ and $i \neq j$ then the probability that $Q(\hat{\imath})$ holds is independent from the probability that $Q(\hat{\jmath})$ holds, that is, $\mathbb{P}_\mu(Q(\hat{\imath}) \wedge Q(\hat{\jmath})) = \mathbb{P}_\mu(Q(\hat{\imath})) \cdot \mathbb{P}_\mu(Q(\hat{\jmath}))$.

(b) If $i, j \in D$ and $i \neq j$ then the probability that $R(\hat{\imath},\hat{\jmath})$ is independent from the probability that $Q(\hat{\imath})$ holds and from the probability that $Q(\hat{\jmath})$ holds, that is, $\mathbb{P}_\mu(R(\hat{\imath},\hat{\jmath}) \wedge Q(\hat{\imath}) \wedge Q(\hat{\jmath})) = \mathbb{P}_\mu(R(\hat{\imath},\hat{\jmath})) \cdot \mathbb{P}_\mu(Q(\hat{\imath})) \cdot \mathbb{P}_\mu(Q(\hat{\jmath}))$.

(c) If $i, j, k, l \in D$ and $i \neq k$ or $j \neq l$, then the probability that $R(\hat{\imath},\hat{\jmath})$ holds is independent from the probability that $R(\hat{k},\hat{l})$ holds, that is, $\mathbb{P}_\mu(R(\hat{\imath},\hat{\jmath}) \wedge R(\hat{k},\hat{l})) = \mathbb{P}_\mu(R(\hat{\imath},\hat{\jmath})) \cdot \mathbb{P}_\mu(R(\hat{k},\hat{l}))$.

It is an elementary, but somewhat tedious, exercise to prove (a)– (c) above so we leave this out. Instead we use (a)–(c)) to compute, or rather estimate, the probability of the sentence

$$\forall x \forall y \exists z \big(Q(z) \wedge R(x,z) \wedge R(z,y)\big)$$

which we denote by φ. We do this by first computing the probability of $\neg\varphi$ and then use the identity $\mathbb{P}_\mu(\varphi) = 1 - \mathbb{P}_\mu(\neg\varphi)$ (which holds since for every $\mathcal{A} \in \mathbf{W}$, $\mathcal{A} \models \varphi$ if and only if $\mathcal{A} \not\models \neg\varphi$).

The sentence $\neg\varphi$ is logically equivalent to

$$\exists x \exists y \forall z \neg\big(Q(z) \wedge R(x,z) \wedge R(z,y)\big)$$

which in turn is logically equivalent to

$$\exists x \exists y \forall z \big(\neg Q(z) \vee \neg R(x,z) \vee \neg R(z,y)\big). \tag{4.2.1}$$

We first aim at computing the probability of the formula (4.2.1).

Take arbitrary $i, j \in D$ and let $\psi(\hat{\imath},\hat{\jmath})$ denote the sentence

$$\forall z \big(\neg Q(z) \vee \neg R(\hat{\imath},z) \vee \neg R(\hat{\jmath},z)\big).$$

We first ask how many $\mathcal{A} \in \mathbf{W}$ there are such that $\mathcal{A} \models \psi(\hat{\imath},\hat{\jmath})$. For every $k \in D$ and $\mathcal{A} \in \mathbf{W}$ we can have $\mathcal{A} \models \neg Q(\hat{k}) \vee \neg R(\hat{\imath},\hat{k}) \vee \neg R(\hat{\jmath},\hat{k})$ because

- $\mathcal{A} \not\models Q(\hat{k})$ or
- $\mathcal{A} \models Q(\hat{k})$ and $\mathcal{A} \not\models R(\hat{\imath},\hat{k}) \vee R(\hat{\jmath},\hat{k})$,

where the two cases, or events, do not overlap. Now we note the following:

(i) The probability that $Q(\hat{k})$ fails for every $k \in D$ is $1/2^n$ by (a) above.

(ii) For every $k \in D$, the probability that $R(\hat{\imath},\hat{k}) \wedge R(\hat{\jmath},\hat{k})$ holds is $1/2 \cdot 1/2 = 1/4$, independently of whether $Q(\hat{k})$ holds or not, by (b) and (c) above. Hence the probability that $\neg R(\hat{\imath},\hat{k}) \vee \neg R(\hat{\jmath},\hat{k})$ holds is $3/4$.

(iii) Conditioned on the event that $Q(\hat{k})$ holds for exactly m elements $k \in D$, it follows from (ii) and (b)–(c) that the probability that $\psi(\hat{\imath},\hat{\jmath})$ holds is $(3/4)^m$.

(iv) For every $1 \le m \le n$ there are $\binom{n}{m}$ ways of choosing a subset of D of size m, that is, $\binom{n}{m}$ ways of choosing an interpretation of Q such that $Q(\hat{k})$ holds for exactly m elements $k \in D$. Then one can choose an interpretation of R in 2^{n^2} ways. It follows that the probability of choosing a structure such that $Q(\hat{k})$ holds for exactly m elements $k \in D$ is $\frac{\binom{n}{m} \cdot 2^{n^2}}{2^{n^2+n}} = \binom{n}{m}/2^n$.

(v) By combining (iii) and (iv), it follows that the probability that $Q(\hat{k})$ holds for exactly m elements $k \in D$ and $\psi(\hat{\imath}, \hat{\jmath})$ holds is $(3/4)^m \binom{n}{m}/2^n$.

From (i) and (v) it follows that the probability that $\psi(\hat{\imath}, \hat{\jmath})$ holds is

$$\begin{aligned}
&1/2^n + \sum_{m=1}^{n} \frac{(3/4)^m \binom{n}{m}}{2^n} = \frac{1}{2^n} + \frac{1}{2^n} \sum_{m=1}^{n} (3/4)^m \binom{n}{m} \le \\
&\frac{1}{2^n} + \frac{1}{2^n} (3/4)^n \sum_{m=1}^{n} \binom{n}{m} = \frac{1}{2^n} + \frac{1}{2^n} (3/4)^n (2^n - 1) = \\
&(1/2)^n + (3/4)^n - (3/8)^n.
\end{aligned}$$

So we see that the probability that $\psi(\hat{\imath}, \hat{\jmath})$ holds is at most $(1/2)^n + (3/4)^n - (3/8)^n$. Since there are n^2 choices for $i, j \in D$ the probability that $\exists x \exists y \psi(x, y)$ holds is at most

$$n^2 \Big((1/2)^n + (3/4)^n - (3/8)^n \Big).$$

But $\exists x \exists y \psi(x, y)$ is logically equivalent to $\neg\varphi$, so the probability that φ holds, that is $\mathbb{P}_\mu(\varphi)$, is at least

$$1 - n^2 \Big((1/2)^n + (3/4)^n - (3/8)^n \Big).$$

Observe that the last expression tends to 1 as n tends to infinity.[4] Moreover, the rate of convergence to 0 is exponential. So if the domain D is large enough we can be almost sure that φ holds. For example, if $n = 50$ we get $\mathbb{P}_\mu(\varphi) \ge 0.9985$.

4.2.9 Example: where we see how a lifted Bayesian network gives rise to a probability distribution on a set of structures

This example is similar to the example in Section 4.2.7 in the sense that it derives a probability distribution on a set of possible worlds, or states, from a lifted Bayesian network. But the situation here is a more complex than that in Section 4.2.7, in spite of the fact that the DAG's of the involved lifted Bayesian networks are the same (up to isomorphism).

Let the domain be $D = \{1, \dots, n\}$. We consider a first-order language L with two relation symbols, Q and R, with arities 1 and 2, respectively, and with a constant symbol $\hat{k}$ for every $k \in D$. Our set of possible worlds, denoted $\mathbf{W}$, is the set of all L-structures with domain D such that for every $k \in \{1, \dots, n\}$, $\hat{k}$ is interpreted as k.

The probabilities that we will assign to structures in $\mathbf{W}$ are determined by the *lifted Bayesian network* in Figure 4.6.

The lifted Bayesian network is interpreted as follows. For any object x the probability that x has the property Q is 1/4. Since x and y are supposed to refer to arbitrary objects and every object has probability 1/4 of having the property Q we must have $w(Q(x)) = w(Q(y))$, that is, for any x and y the probability that x has the property Q is the same as the probability

[4]Because if $P(x)$ is a polynomial and $0 < a < 1$, then $\lim_{x\to\infty} P(x)a^x = 0$.

Figure 4.6:

$$\begin{tikzcd}
Q(x) \arrow[dr] & & Q(y) \arrow[dl] \\
& R(x,y) &
\end{tikzcd}$$

$w(Q(x)) = w(Q(y)) = 1/4$	$w(R(x,y) \vert Q(x) \wedge Q(y)) = 1/2$
	$w(R(x,y) \vert Q(x) \wedge \neg Q(y)) = 1/4$
	$w(R(x,y) \vert \neg Q(x) \wedge Q(y)) = 1/4$
	$w(R(x,y) \vert \neg Q(x) \wedge \neg Q(y)) = 1/8$

that y has the property Q. Since there is no arrow between $Q(x)$ and $Q(y)$ the network tells that if $x \neq y$ then the probability that x has property Q is independent of whether y has property Q or not, and vice versa. Then we have conditional probabilities which describe how the probability of $R(x,y)$ (i.e. that x and y satisfy the relation $R(x,y)$) depends on the whether $Q(x)$ and/or $Q(y)$ hold or not. For example, the first conditional probability tells that, if we know that both x and y have the property Q, then the probability that $R(x,y)$ holds is $1/2$. The second conditional probability tells that, if we know that x has the property Q and that y does not have the property Q, then the probability that $R(x,y)$ holds is $1/4$. And so on.

Now we think about the objects to which x and y refer as being members of the domain $D = \{1, \ldots, n\}$. Then the lifted Bayesian network can be seen as a procedure to construct, by probabilistic means, a structure with domain D as follows:

1. Take a biased coin which has probability $1/4$ of ending up with a head. For each $i \in D$, toss the coin. If it comes up with a head then i is included in the interpretation of Q, in other words, $Q(\hat{\imath})$ holds in this structure; otherwise $Q(\hat{\imath})$ does not hold.

2. Take a biased coin which has probability $1/2$ of ending up with a head. For every pair $(i,j) \in D^2$ such that $Q(\hat{\imath}) \wedge Q(\hat{\jmath})$ holds, toss the coin. If it comes up with a head then (i,j) is included in the interpretation of R, in other words, $R(\hat{\imath},\hat{\jmath})$ holds in this structure; otherwise $R(\hat{\imath},\hat{\jmath})$ does not hold.

3. Take a biased coin which has probability $1/4$ of ending up with a head. For every pair $(i,j) \in D^2$ such that $Q(\hat{\imath}) \wedge \neg Q(\hat{\jmath})$ holds, toss the coin. If it comes up with a head then (i,j) is included in the interpretation of R, in other words, $R(\hat{\imath},\hat{\jmath})$ holds in this structure; otherwise $R(\hat{\imath},\hat{\jmath})$ does not hold.

4. Like the previous step but change '$Q(\hat{\imath}) \wedge \neg Q(\hat{\jmath})$' to '$\neg Q(\hat{\imath}) \wedge Q(\hat{\jmath})$'.

5. Like the previous step but change the probability of the coin ending up with a head to $1/8$ and change '$\neg Q(\hat{\imath}) \wedge Q(\hat{\jmath})$' to '$\neg Q(\hat{\imath}) \wedge \neg Q(\hat{\jmath})$'.

Observe the following: The probability that $Q(\hat{\imath})$ holds for exactly q elements $i \in D$ is $(1/4)^q(3/4)^{n-q}$. Conditioned on this, the probability that $R(\hat{\imath},\hat{\jmath}) \wedge Q(\hat{\imath},\hat{\jmath})$ holds for exactly r_1 pairs $(i,j) \in D^2$ is $(1/4)^{r_2}(3/4)^{q(n-q)-r_2}$. Conditioned on the event that $Q(\hat{\imath})$ holds for exactly q elements $i \in D$, the probability that $R(\hat{\imath},\hat{\jmath}) \wedge \neg Q(\hat{\imath},\hat{\jmath})$ holds for exactly r_2 pairs $(i,j) \in D^2$ is $(1/4)^{r_2}(3/4)^{q(n-q)-r_2}$. And so on.

The idea with this construction is that if we construct a structure in the described way (and D's size is large), then we are likely to get a structure for which the dependencies and (conditional) probabilities of the lifted Bayesian network are quite good approximations (the larger the size of D, then better the approximation). Using the same idea as in 1–5 above we define a probability distribution, also denoted w, on $\mathbf{W}$ as follows:

For every $\mathcal{A} \in \mathbf{W}$, if

$$\begin{aligned} q = &\text{ the number of } i \in D \text{ such that } \mathcal{A} \models Q(\hat{\imath}), \\ r_1 = &\text{ the number of pairs } (i,j) \in D^2 \text{ such that } \mathcal{A} \models R(\hat{\imath},\hat{\jmath}) \wedge Q(\hat{\imath}) \wedge Q(\hat{\jmath}), \\ r_2 = &\text{ the number of pairs } (i,j) \in D^2 \text{ such that } \mathcal{A} \models R(\hat{\imath},\hat{\jmath}) \wedge Q(\hat{\imath}) \wedge \neg Q(\hat{\jmath}), \\ r_3 = &\text{ the number of pairs } (i,j) \in D^2 \text{ such that } \mathcal{A} \models R(\hat{\imath},\hat{\jmath}) \wedge \neg Q(\hat{\imath}) \wedge Q(\hat{\jmath}) \text{ and} \\ r_4 = &\text{ the number of pairs } (i,j) \in D^2 \text{ such that } \mathcal{A} \models R(\hat{\imath},\hat{\jmath}) \wedge \neg Q(\hat{\imath}) \wedge \neg Q(\hat{\jmath}), \end{aligned}$$

then

$$\begin{aligned} w(\mathcal{A}) = &(1/4)^q (3/4)^{n-q} (1/2)^{r_1} (1/2)^{q^2 - r_1} (1/4)^{r_2} (3/4)^{q(n-q)-r_2} \\ &(1/4)^{r_3} (3/4)^{(n-q)q - r_3} (1/8)^{r_4} (7/8)^{(n-q)^2 - r_4}. \end{aligned}$$

We extend w to the set of subsets of $\mathbf{W}$ in the usual way:

$$\text{For every } \mathbf{X} \subseteq \mathbf{W}, \text{ let } w(\mathbf{X}) = \sum_{\mathcal{A} \in \mathbf{X}} w(\mathcal{A}).$$

For every L-sentence φ, we define "the probability that φ is true in a structure (or world) in $\mathbf{W}$", denoted $\mathbb{P}_w(\varphi)$, by

$$\mathbb{P}_w(\varphi) = w\big(\{\mathcal{A} \in \mathbf{W} : \mathcal{A} \models \varphi\}\big).$$

For sentences φ and ψ, we let

$$\mathbb{P}_w(\varphi|\psi) = \frac{\mathbb{P}_w(\varphi \wedge \psi)}{\mathbb{P}_w(\psi)}.$$

So $\mathbb{P}_w(\varphi|\psi)$ is the conditional probability that φ is true, assuming that that ψ is true. One can show that the following holds for all $i, j \in D$: [5]

$$\begin{aligned} &\mathbb{P}_w(Q(\hat{\imath})) = 1/4, \\ &\mathbb{P}_w(Q(\hat{\imath}) \wedge Q(\hat{\jmath})) = \mathbb{P}_w(Q(\hat{\imath})) \cdot \mathbb{P}_w(Q(\hat{\jmath})) \quad \text{if } i \neq j, \\ &\mathbb{P}_w(R(\hat{\imath},\hat{\jmath})|\ Q(\hat{\imath}) \wedge Q(\hat{\jmath})) = 1/2, \\ &\mathbb{P}_w(R(\hat{\imath},\hat{\jmath})|\ Q(\hat{\imath}) \wedge \neg Q(\hat{\jmath})) = 1/4, \\ &\mathbb{P}_w(R(\hat{\imath},\hat{\jmath})|\ \neg Q(\hat{\imath}) \wedge Q(\hat{\jmath})) = 1/4, \\ &\mathbb{P}_w(R(\hat{\imath},\hat{\jmath})|\ \neg Q(\hat{\imath}) \wedge \neg Q(\hat{\jmath})) = 1/8. \end{aligned}$$

Moreover, in the last four lines above, the probability is independent of whether $R(\hat{k},\hat{l})$ holds or not if $k \neq i$ or $l \neq j$. This means that the function $\mathbb{P}_w$ "matches" the lifted Bayesian network which we started with, in the following sense: The lifted Bayesian network

[5] Again a straighforward but tedious exercise.

expresses that "for every object x the probability that x has the property Q is 1/4", while "for every $i \in D$, $\mathbb{P}_w(\hat{\imath}) = 1/4$" says that, for every object in the domain D, the probability that it has the property Q (or "satisfies Q") is 1/4. The matching is similar for the conditional probabilities of the Bayesian network and for distinct $i, j \in D$, the probability that $Q(\hat{\imath})$ holds is independent from the probability that $Q(\hat{\jmath})$ holds.

Now we compute the probability of a more complex sentence. Let $\chi(x, y)$ denote the formula

$$\exists z\big(z \neq x \wedge z \neq y \wedge Q(z) \wedge R(x,z) \wedge R(z,y)\big)$$

and let θ denote the sentence

$$\forall x \forall y\big(x \neq y \rightarrow \chi(x,y)\big).$$

Let $i, j \in D$ and $i \neq j$. We first compute the probability that $\chi(\hat{\imath}, \hat{\jmath})$ holds.

- The probability that $Q(\hat{\imath}) \wedge Q(\hat{\jmath})$ holds is $(1/4)(1/4) = 1/16$. Assume that $Q(\hat{\imath}) \wedge Q(\hat{\jmath})$ holds. For every $k \in D$ different from both i and j, the probability that $Q(\hat{k}) \wedge R(\hat{\imath}, \hat{k}) \wedge R(\hat{k}, \hat{\jmath})$ holds is $(1/4)(1/2)(1/2) = 1/16$, so the probablity that $R(\hat{\imath}, \hat{k}) \wedge R(\hat{k}, \hat{\jmath})$ fails is $1 - 1/16 = 15/16$. As there are $n-2$ candidates for k, the probability that $\chi(\hat{\imath}, \hat{\jmath})$ fails is $(15/16)^{n-2}$. Hence the probability that $\chi(\hat{\imath}, \hat{\jmath})$ holds is $1 - (15/16)^{n-2}$. Thus, no longer assuming that $Q(\hat{\imath}) \wedge Q(\hat{\jmath})$ holds, the probability that $Q(\hat{\imath}) \wedge Q(\hat{\jmath}) \wedge R(\hat{\imath}, \hat{k}) \wedge R(\hat{k}, \hat{\jmath})$ holds is $\frac{1}{16}\big(1 - (15/16)^{n-2}\big)$.

- By similar reasoning as in the previous case, the probability that $Q(\hat{\imath}) \wedge \neg Q(\hat{\jmath}) \wedge R(\hat{\imath}, \hat{k}) \wedge R(\hat{k}, \hat{\jmath})$ holds is $\frac{3}{16}\big(1 - (31/32)^{n-2}\big)$.

- By similar reasoning again, the probability that $\neg Q(\hat{\imath}) \wedge Q(\hat{\jmath}) \wedge R(\hat{\imath}, \hat{k}) \wedge R(\hat{k}, \hat{\jmath})$ holds is $\frac{3}{16}\big(1 - (31/32)^{n-2}\big)$.

- By similar reasoning yet again, the probability that $\neg Q(\hat{\imath}) \wedge \neg Q(\hat{\jmath}) \wedge R(\hat{\imath}, \hat{k}) \wedge R(\hat{k}, \hat{\jmath})$ holds is $\frac{9}{16}\big(1 - (63/64)^{n-2}\big)$.

It follows that the probability that $\chi(\hat{\imath}, \hat{\jmath})$ holds is

$$\begin{aligned} &\frac{1}{16}\big(1 - (15/16)^{n-2}\big) \;+\; 2 \cdot \frac{3}{16}\big(1 - (31/32)^{n-2}\big) \;+\; \frac{9}{16}\big(1 - (63/64)^{n-2}\big) \\ &\geq \Big(\frac{1}{16} + \frac{6}{16} + \frac{9}{16}\Big)\Big(1 - (63/64)^{n-2}\Big) = 1 - (63/64)^{n-2}. \end{aligned}$$

Hence the probability that $\chi(\hat{\imath}, \hat{\jmath})$ fails is at most $(63/64)^{n-2}$. The sentence θ fails if $\chi(\hat{\imath}, \hat{\jmath})$ fails for at least one choice of distinct $i, j \in D$. There are $n(n-1)$ choices of distinct $i, j \in D$, so the probability that θ *fails* is at most $n(n-1)(63/64)^{n-2}$. Hence the probability that θ holds is at least $1 - n(n-1)(63/64)^{n-2}$. We have $n(n-1)(63/64)^{n-2} \to 0$ as $n \to \infty$. So if the domain is large θ holds with high probability, that is, $\mathbb{P}_w(\theta)$ is close to 1. If $n = 1000$ we get $\mathbb{P}_w(\theta) \geq 0.85$ and if $n = 2000$ we get $\mathbb{P}_w(\theta) \geq 0.9999999$. (Convergence to 1 is fast once it "catches up", because the convergence has exponential speed.)

4.2.10 General definitions of probabilities in the the "possible worlds" approach for first-order logic

The definitions in the examples in Section 4.2.8 and 4.2.9 are instances of the general definitions formulated in this section. The assumptions and definitions in this section follow the same pattern as in Section 4.2.5 *but* in this section the "possible worlds" are *first-order structures* (not truth assignments) and we consider *first-order formulas* instead of propositional formulas.

Let L be a first-order language (that is, the set of first-order formulas which can be constructed using a specified set of relation symbols, function symbols and constant symbols). Let D be a set and let $\mathbf{W}$ be a finite or countable set of L-structures with domain D (that is, first-order structures in which the relation symbols, function symbols and constant symbols have interpretations in D). Furthermore, suppose that $\mu : \mathbf{W} \to \mathbb{R}$ is a probability distribution.

The probability distribution $\mu : \mathbf{W} \to \mathbb{R}$ can be extended to a *probability measure* on the set of all subsets of $\mathbf{W}$, which we give the same name μ, as follows:

$$\text{For every } \mathbf{X} \subseteq \mathbf{W}, \mu(\mathbf{X}) := \sum_{\mathcal{A} \in \mathbf{X}} \mu(\mathcal{A}).$$

For every L-sentence (i.e. L-formula without free variables) φ we define

$$\mathbb{P}_\mu(\varphi) := \mu\big(\{\mathcal{A} \in \mathbf{W} : \mathcal{A} \models \varphi\}\big)$$

and for all L-sentences φ and ψ we define

$$\mathbb{P}_\mu(\varphi \mid \psi) := \frac{\mathbb{P}_\mu(\varphi \wedge \psi)}{\mathbb{P}_\mu(\psi)}.$$

We read '$\mathbb{P}_\mu(\varphi)$' as "the probability of φ" or "the probability that φ is true", or something similar. The notation "$\mathbb{P}_\mu(\varphi \mid \psi)$" is read as "the (conditional) probability that φ is true given that ψ is true", or something similar. In the present setting of first-order logic we have an analogue of Proposition 4.2.1 but here φ and ψ denote first-order sentences and $\mathbf{W}$ is a set of first-order structures.

Proposition 4.2.5 (Basic properties of $\mathbb{P}_\mu$)

(a) If φ and ψ are logically equivalent sentences, then $\mathbb{P}_\mu(\varphi) = \mathbb{P}_\mu(\psi)$.

(b) $\mathbb{P}_\mu(\neg\varphi) = 1 - \mathbb{P}_\mu(\varphi)$ for all sentences φ.

(c) Suppose that for every $\mathcal{A} \in \mathbf{W}$, $\mathcal{A} \not\models \varphi \wedge \psi$, in other words that there is no "world" in which both φ and ψ are true. Then $\mathbb{P}_\mu(\varphi \vee \psi) = \mathbb{P}_\mu(\varphi) + \mathbb{P}_\mu(\psi)$.

(d) If is a valid sentence (that is, a sentence which is true in every structure), then $\mathbb{P}_\mu(\varphi) = 1$.

Remark 4.2.6 Suppose that $\mathbb{P}_\mu(\varphi) = 1$. Even if $\mathbf{W}$ contains all structures with domain D (for the given first-order language L) we cannot, in general, conclude that φ is valid. The reason is that there may be some structure with a different domain than D (perhaps having a larger or smaller domain) in which φ is false.

Remark 4.2.7 (The probability of logical implication is *not* the same as conditional probability) Just as I pointed out in Remark 4.2.3 for propositional logic, the equality $\mathbb{P}_\mu(\psi \to \varphi) = \mathbb{P}_\mu(\varphi \mid \psi)$ does *not* hold in general for first-order logic. The reason for this is essentially the same as in the case of propositional logic. The difference is just that in the context of first-order logic $\mathbf{W}$ is a set of first-order structures and φ and ψ first-order sentences.

4.3 Probabilities on a domain, or the statistical information approach

In the approach to logic and probability taken in this chapter we do *not* consider a set of possible worlds and a probability distribution this set. Instead we consider *only one* world, which formally speaking will be an L-structure, say $\mathcal{A}$ with domain A, for some first-order language L. The probabilistic viewpoint comes in via introducing a probability distribution μ on the domain A. For every $n > 1$, μ induces the product distribution μ^n on A^n as explained in Section 4.1. Using μ^n we can define, for every L-formula $\varphi(x_1, \ldots, x_n)$, "the probability that $\varphi(x_1, \ldots, x_n)$ is satisfied (in $\mathcal{A}$) by a randomly chosen n-tuple $(a_1, \ldots, a_n) \in A^n$". Since "the probabilities on a domain approach" requires that we have a domain of objects, this approach does not make sense for propositional logic, so in this section we only use first-order logic as our modelling language.

In Section 4.3.1 the ideas are illustrated by means of an example. The general definitions and some general results are given in Section 4.3.2. In Section 4.3.3 another example is given which is superficially similar to the example in Section 4.2.6, but one should observe that different approaches to "logic and probability" are used in Sections 4.2.6 and 4.3.3.

Note: *As in previous sections the examples are an important and integrated part of this text and should be read as carefully as the theoretical parts.*

4.3.1 Example: statistical information of a database

Let us consider a database which relates certain people with certain occupations as illustrated in the following table (which we interpret as saying that Greta has two jobs and Hans none):

person	occupation
Anna	teacher
Bengt	carpenter
Cecilia	driver
David	teacher
Eva	nurse
Fredrik	driver
Greta	carpenter, driver
Hans	–

For the population of people occuring in the database we may want to ask questions like "What is the likelihood that a randomly chosen person has a certain occupation?" or "What is the likelihood that two randomly chosen persons have the same occupation?". In other words we may be interested in the probability that (for example) the following statement is true for a randomly chosen person x from the database: "x is a teacher". The syntax of some version of the *structured query language (SQL)* can be used to express statements such as "x is a teacher" or "x and y have the same occupation". Here we will use first-order logic to express such statements as "x is a teacher". In order to do this we need to represent the database as a first-order structure. There is not a unique way to do this. Different approaches may have different advantages and disadvantages.

Our first approach is to assing a unary relation symbol to each occupation: T for "teacher", C for "carpenter", D for "driver" and N for "nurse". The first-order language that we consider have only these relation symbols. Let

$$A = \{Anna, Bengt, Cecilia, David, Eva, Fredrik, Greta, Hans\}$$

be the domain of the structure that will represent the database. The structure itself (which also has information about the interpretation of the relation symbols) is called $\mathcal{A}$. In $\mathcal{A}$ the relation symbols are interpreted as follows:

$$\begin{aligned} T^{\mathcal{A}} &= \{Anna, David\}, \\ C^{\mathcal{A}} &= \{Bengt, Greta\}, \\ D^{\mathcal{A}} &= \{Cecilia, Fredrik, Greta\}, \\ N^{\mathcal{A}} &= \{Eva\}. \end{aligned}$$

As usual we assume that we have a constant symbol for every element in the domain which is interpreted as this element. In other words we have constant symbols $\widehat{Anna}, \widehat{Bengt}, \dots$ where $\widehat{Anna}$ is interpreted as $Anna$, $\widehat{Bengt}$ is interpreted as $Bengt$, and so on. In the structure $\mathcal{A}$ the statement "x is a teacher" is expressed by the first-order formula $T(x)$. The statement "x and y are different and have the same occupation" is expressed by

$$x \neq y \wedge \big((T(x) \wedge T(y)) \vee (C(x) \wedge C(y)) \vee (D(x) \wedge D(y)) \vee (N(x) \wedge N(y))\big).$$

The statement "x is a teacher and there is someone else who is a teacher" is expressed by $T(x) \wedge \exists y(y \neq x \wedge T(y))$.

If we assume that each person in the domain A is equaly likely to be chosen if we "randomly choose" someone, then we are in fact using the uniform probability measure on A. So let $\mu : A \to \mathbb{R}$ the uniform probability measure. This means that $\mu(a) = 1/8$ for every $a \in A$ (since $|A| = 8$). Recall from Section 4.1 that μ induces a product measure μ^n on A^n for every $n > 1$. For every $n > 0$ and every first-order formula $\varphi(x_1, \dots, x_n)$ (with free variables $x_1, \dots, x_n$) define

$$\mathbb{P}_\mu\big(\varphi(x_1, \dots, x_n)\big) = \mu^n\big(\{(a_1, \dots, a_n) \in A^n : \mathcal{A} \models \varphi(\hat{a}_1, \dots, \hat{a}_n)\}\big).$$

Here we identify A^1 with A and μ^1 with μ. Then $\mathbb{P}_\mu\big(\varphi(x_1, \dots, x_n)\big)$ is the *probability that a randomly chosen n-tupe* $(a_1, \dots, a_n)$ *will satisfy the formula* $\varphi(x_1, \dots, x_n)$". For example, the probability that a randomly chosen element of A (a person that is) is a teacher is

$$\mathbb{P}_\mu(T(x)) = \mu\big(\{Anna, David\}\big) = \mu(Anna) + \mu(David) = 1/8 + 1/8 = 1/4.$$

Let $\varphi(x, y)$ be the formula above which expresses that "x and y have the same occupation". The probability that randomly chosen (not necessarily different) x and y will be different and have the same occupation is

$$\begin{aligned} \mathbb{P}_\mu(\varphi(x, y)) = \mu^2\big(\{&(Anna, David), (Bengt, Greta), (Cecilia, Fredrik), \\ &(Cecilia, Greta), (Fredrik, Greta)\}\big) = \\ &\mu^2(Anna, David) + \mu^2(Bengt, Greta) + \mu^2(Cecilia, Fredrik) \\ &+\mu^2(Cecilia, Greta) + \mu^2(Fredrik, Greta) = 5 \cdot \frac{1}{8} \cdot \frac{1}{8} = 5/64. \end{aligned}$$

Note that 10/64 is the proportion of 2-tuples $(a, b) \in A^2$ which satisfy $\varphi(x, y)$ (as $|A^2| = 8^2 = 64$). It is perhaps more natural to consider the probability that "x and y have the same occupation, *assuming* that x and y are *different* persons". This is given by the conditional probability

$$\mathbb{P}_\mu(\varphi(x, y) | x \neq y) = \frac{\mathbb{P}_\mu(\varphi(x, y) \wedge x \neq y)}{\mathbb{P}_\mu(x \neq y)} = \frac{10/64}{8 \cdot 7/64} = \frac{5}{28}.$$

In some situations the uniform probability distribution μ on A may not be appropriate. Let's suppose that "recruiters" looking for "the right person for the right job" are lazy and tend to look at people from the beginning of the alphabet more often than people later in the alphabet. Let $\lambda(Anna) = 8/36$, $\lambda(Bengt) = 2/36$, $\lambda(Cecilia) = 3/36$, $\ldots$, $\lambda(Hans) = 1/36$. Then λ is a probability distribution on A. With $\mathbb{P}_\lambda$ defined analogously as $\mathbb{P}_\mu$ we get

$$\mathbb{P}_\lambda(T(x)) = \lambda\big(\{Anna, David\}\big) = \mu(Anna) + \mu(David) = 8/36 + 5/36 = 13/36.$$

and

$$\begin{aligned}\mathbb{P}_\lambda(\varphi(x,y)) &= \lambda^2\big(\{(Anna, David), (Bengt, Greta), (Cecilia, Fredrik),\\ &\qquad (Cecilia, Greta), (Fredrik, Greta)\}\big)\\ &= \frac{8}{36}\cdot\frac{5}{36} + \frac{7}{36}\cdot\frac{2}{36} + \frac{6}{36}\cdot\frac{3}{36} + \frac{6}{36}\cdot\frac{2}{36} + \frac{3}{36}\cdot\frac{2}{36}.\end{aligned}$$

A disadvantage with the structure $\mathcal{A}$ is that if if new occupations are added then we need to expand the first-order language with new relation symbols (for these new occupations). Also, sometimes first-order formulas which express simple properties or relations become rather long. Think of the formula above which expresses that x and y have the same occupation (and what if there where, for example, 1000 occupations). Now we describe a different way of representing the database by a first-order structure. The domain of the structure is

$$\begin{aligned}B = \{&Anna, Bengt, Cecilia, David, Eva, Fredrik, Greta, Hans,\\ &teacher, carpenter, driver, nurse\},\end{aligned}$$

so both persons and occupations are now included in the domain. We would like to be able to distingish between persons and occupations so we use two unary relations symbols P and O. The formulas $P(x)$ and $O(x)$ are intended to express that "x is a person" and "x is an occupation", respectively. We also want to be able to tell what occupation any given person has, so we use a binary relation symbol W for this: $W(x,y)$ expresses that "x has occupation y". We call the new structure $\mathcal{B}$ and in this structure we have the following interpretations:

$$\begin{aligned}P^{\mathcal{B}} &= \{Anna, Bengt, Cecilia, David, Eva, Fredrik, Greta, Hans\},\\ O^{\mathcal{B}} &= \{teacher, carpenter, driver, nurse\},\\ W^{\mathcal{B}} &= \{(Anna, teacher), (Bengt, carpenter), (Cecilia, driver), (David, teacher),\\ &\qquad (Eva, nurse), (Fredrik, driver), (Greta, carpenter), (Greta, driver)\}.\end{aligned}$$

With this setup we can add new persons and new occupations to the domain without changing the first-order language that we use. But to express that a person has a particular occupation we need to use a constant symbol refering to that occupation. So, for example, the formula $W(x, \widehat{teacher})$ expresses that "x is a teacher" where $\widehat{teacher}$ is a constant symbol which (in $\mathcal{B}$) is interpreted as *teacher*. We can easily express that "x and y have the same occupation" with the formula $\exists z(W(x,z) \wedge W(y,z))$.

Let γ be the uniform probability distribution on B, that is, $\gamma(b) = 1/12$ for every $b \in B$. Suppose that we want to compute the probability that a randomly chosen *person* is a teacher. We have $\mathbb{P}_\gamma(W(x, \widehat{teacher}) = \gamma(\{Anna, David\}) = 1/12 + 1/12 = 1/6$. *But* $\mathbb{P}_\gamma(W(x, \widehat{teacher})$ *is the probability that a randomly chosen element from* B *satisfies*

$W(x, \widehat{teacher})$ and there are elements in B (such as carpenter) which are not persons but occupations. So we need to consider a conditional probability. In other words we compute the probability that x is a teacher, conditioned on the assumption that x is a person. This probability is

$$\mathbb{P}_\gamma\big(W(x, \widehat{teacher}) \bigm| P(x)\big) = \frac{\mathbb{P}_\gamma\big(W(x, \widehat{teacher})\big) \wedge P(x)\big)}{\mathbb{P}_\gamma\big(P(x)\big)} = \frac{2/12}{8/12} = 1/4.$$

Thinking in the same way we have that the probability that x and y have the same occupation, conditioned on the assumption that x and y are different persons, is

$$\mathbb{P}_\gamma\big(\exists z(W(x,z) \wedge W(y,z)) \bigm| P(x) \wedge P(y) \wedge x \neq y\big) =$$
$$\frac{\mathbb{P}_\gamma\big(\exists z(W(x,z) \wedge W(y,z)) \wedge P(x) \wedge P(y) \wedge x \neq y\big)}{\mathbb{P}_\gamma\big(P(x) \wedge P(y) \wedge x \neq y\big)} = \frac{10/12^2}{8 \cdot 7/12^2} = \frac{5}{28}.$$

Observe that we get the same probability as in $\mathcal{A}$ with $\mathbb{P}_\mu$ for "x is a teacher" and for "x and y are different persons with the same occupation". This should be expected because in both cases we use the uniform probability distribution on the set of persons.

4.3.2 General definitions in the context of probabilities on domains

For all of this section we let L be a first-order language and $\mathcal{M}$ an L-structure with domain M. We assume that for every $a \in M$ there is a constant symbol $\hat{a}$ which is interpreted as a (or with symbols, such that $\hat{a}^{\mathcal{M}} = a$). Furthermore, we let $\mu : M \to \mathbb{R}$ be a probability distribution. Recall from Section 4.1 μ induces a probability distribution $\mu^n : M^n \to \mathbb{R}$ for every natural number $n > 0$ (and we indentify M^1 and μ^1 with M and μ, respectively). When we say 'formula' in this section we mean (of course) 'L-formula'.

Definition 4.3.1 For every formula $\varphi(x_1, \ldots, x_n)$ such that all free variables of it belong to the sequence $x_1, \ldots, x_n$ we define "the probability that $\varphi(x_1, \ldots, x_n)$ is satisfied by an n-tuple from M", denoted $\mathbb{P}_\mu\big(\varphi(x_1, \ldots, x_n)\big)$, by

$$\mathbb{P}_\mu^{\mathcal{M}}\big(\varphi(x_1, \ldots, x_n)\big) = \mu^n\big(\{(a_1, \ldots, a_n) \in M^n : \mathcal{M} \models \varphi(\hat{a}_1, \ldots, \hat{a}_n)\}\big).$$

Hence '$\mathbb{P}_\mu^{\mathcal{M}}\big(\varphi(x_1, \ldots, x_n)\big)$' is just another notation for

$$\mu^n\big(\{(a_1, \ldots, a_n) \in M^n : \mathcal{M} \models \varphi(\hat{a}_1, \ldots, \hat{a}_n)\}\big)$$

but it is a convenient notation to have.

Remark 4.3.2 Observe that in Definition 4.3.1 we consider formulas with free variables among $x_1, \ldots, x_n$. *Suppose that $\varphi(x_1, \ldots, x_n)$ denotes a sentence* which means that none of $x_1, \ldots, x_n$ occurs as a free variable in the formula denoted by $\varphi(x_1, \ldots, x_n)$. Then, for any $(a_1, \ldots, a_n) \in M^n$, the substitution whereby each free occurence of x_i in $\varphi(x_1, \ldots, x_n)$ is replaced by $\hat{a}_i$ (for $i = 1, \ldots, n$) does not change the formula. In other words, for all $(a_1, \ldots, a_n), (b_1, \ldots, b_n) \in M^n$, $\varphi(\hat{a}_1, \ldots, \hat{a}_n)$ and $\varphi(\hat{b}_1, \ldots, \hat{b}_n)$ denote the very same sentence. So either both are true in $\mathcal{M}$ or both are false in $\mathcal{M}$. The conclusion is:

> *If $\varphi(x_1, \ldots, x_n)$ denotes a sentence (i.e. a formula without free variables), then either $\mathbb{P}_\mu^{\mathcal{M}}\big(\varphi(x_1, \ldots, x_n)\big) = 1$ or $\mathbb{P}_\mu^{\mathcal{M}}\big(\varphi(x_1, \ldots, x_n)\big) = 0$, depending on whether this sentence is true or false in $\mathcal{M}$.*

In analogy with Propositions 4.2.1 and 4.2.5 we have the following:

Proposition 4.3.3 (Basic properties of $\mathbb{P}_\mu^{\mathcal{M}}$)

(a) If $\mathcal{M} \models \forall x_1 \ldots \forall x_n\big(\varphi(x_1,\ldots,x_n) \leftrightarrow \psi(x_1,\ldots,x_n)\big)$ then $\mathbb{P}_\mu^{\mathcal{M}}\big(\varphi(x_1,\ldots,x_n)\big) = \mathbb{P}_\mu^{\mathcal{M}}\big(\psi(x_1,\ldots,x_n)\big)$. (Thus the conclusion holds in particular if $\varphi(x_1,\ldots,x_n)$ and $\psi(x_1,\ldots,x_n)$ are logically equivalent.)

(b) $\mathbb{P}_\mu^{\mathcal{M}}\big(\neg\varphi(x_1,\ldots,x_n)\big) = 1 - \mathbb{P}_\mu^{\mathcal{M}}\big(\varphi(x_1,\ldots,x_n)\big)$ for all formulas $\varphi(x_1,\ldots,x_n)$.

(c) If $\mathcal{M} \models \neg\exists x_1 \ldots \exists x_n\big(\varphi(x_1,\ldots,x_n) \wedge \psi(x_1,\ldots,x_n)\big)$, then

$$\mathbb{P}_\mu^{\mathcal{M}}\big(\varphi(x_1,\ldots,x_n) \vee \psi(x_1,\ldots,x_n)\big) = \mathbb{P}_\mu^{\mathcal{M}}\big(\varphi(x_1,\ldots,x_n)\big) + \mathbb{P}_\mu^{\mathcal{M}}\big(\psi(x_1,\ldots,x_n)\big).$$

Lemma 4.3.4 (About adding "dummy variables") *Let $\varphi(x_1,\ldots,x_k)$ be a formula (so every free variable that occurs in φ belongs to the sequence $x_1,\ldots,x_k$). Let $n > k$ and let $\varphi'(x_1,\ldots,x_n)$ be the same formula as $\varphi(x_1,\ldots,x_k)$ but view $\varphi'(x_1,\ldots,x_n)$ as a formula in the free variables $x_1,\ldots,x_n$. Alternatively, let $\varphi'(x_1,\ldots,x_n)$ be the formula*

$$\varphi(x_1,\ldots,x_k) \wedge (x_{k+1} = x_{k+1}) \wedge \ldots \wedge (x_n = x_n).$$

(Either way, for all $(a_1,\ldots,a_n) \in M^n$, $\mathcal{M} \models \varphi'(\hat{a}_1,\ldots,\hat{a}_n)$ if and only if $\mathcal{M} \models \varphi(\hat{a}_1,\ldots,\hat{a}_k)$.) Then $\mathbb{P}_\mu^{\mathcal{M}}\big(\varphi'(x_1,\ldots,x_n)\big) = \mathbb{P}_\mu^{\mathcal{M}}\big(\varphi(x_1,\ldots,x_k)\big)$.

Proof. For every $m > 0$, $\mu^m : M^m \to \mathbb{R}$ (where $\mu^m(a_1,\ldots,a_m) = \mu(a_1)\cdot\ldots\cdot\mu(a_n)$) is a probability distribution, so $\sum_{(a_1,\ldots,a_m)\in M^m} \mu^m(a_1,\ldots,a_m) = 1$. This will be used below.

$$\begin{aligned}
&\mathbb{P}_\mu^{\mathcal{M}}\big(\varphi'(x_1,\ldots,x_n)\big) = \\
&\mu^n\big(\{(a_1,\ldots,a_n) \in M^n : \mathcal{M} \models \varphi'(\hat{a}_1,\ldots,\hat{a}_n)\}\big) = \\
&\mu^n\big(\{(a_1,\ldots,a_n) \in M^n : \mathcal{M} \models \varphi(\hat{a}_1,\ldots,\hat{a}_k)\}\big) = \\
&\sum_{\substack{(a_1,\ldots,a_n)\in M^n \\ \mathcal{M}\models\varphi(\hat{a}_1,\ldots,\hat{a}_k)}} \mu^n(a_1,\ldots,a_n) \;=\; \sum_{\substack{(a_1,\ldots,a_n)\in M^n \\ \mathcal{M}\models\varphi(\hat{a}_1,\ldots,\hat{a}_k)}} \mu(a_1)\cdot\ldots\cdot\mu(a_n) = \\
&\sum_{\substack{(a_1,\ldots,a_k)\in M^k \\ \mathcal{M}\models\varphi(\hat{a}_1,\ldots,\hat{a}_k)}} \mu(a_1)\cdot\ldots\cdot\mu(a_k)\Bigg(\sum_{(a_{k+1},\ldots,a_n)\in M^{n-k}} \mu(a_{k+1})\cdot\ldots\cdot\mu(a_n)\Bigg) = \\
&\sum_{\substack{(a_1,\ldots,a_k)\in M^k \\ \mathcal{M}\models\varphi(\hat{a}_1,\ldots,\hat{a}_k)}} \mu^k(a_1,\ldots,a_k)\Bigg(\sum_{(a_{k+1},\ldots,a_n)\in M^{n-k}} \mu^{n-k}(a_{k+1},\ldots,a_n)\Bigg) = \\
&\sum_{\substack{(a_1,\ldots,a_k)\in M^k \\ \mathcal{M}\models\varphi(\hat{a}_1,\ldots,\hat{a}_k)}} \mu^k(a_1,\ldots,a_k) = \\
&\mu^k\big(\{(a_1,\ldots,a_k) \in M^k : \mathcal{M} \models \varphi(\hat{a}_1,\ldots,\hat{a}_k)\}\big) = \\
&\mathbb{P}_\mu^{\mathcal{M}}\big(\varphi(x_1,\ldots,x_k)\big).
\end{aligned}$$

□

Definition 4.3.5 Under the same assumptions as in Definition 4.3.1, but also adding the assumption that $\psi(x_1, \dots, x_n)$ is a formula, we define

$$\mathbb{P}_\mu^{\mathcal{M}}\big(\varphi(x_1, \dots, x_n) \mid \psi(x_1, \dots, x_n)\big) = \frac{\mathbb{P}_\mu^{\mathcal{M}}\big(\varphi(x_1, \dots, x_n) \wedge \psi(x_1, \dots, x_n)\big)}{\mathbb{P}_\mu^{\mathcal{M}}\big(\psi(x_1, \dots, x_n)\big)},$$

which we read as "the conditional probability of $\varphi(x_1, \dots, x_n)$ given (or assuming) $\psi(x_1, \dots, x_n)$".

Remark 4.3.6 (Logical implication is *not* the same as conditional probability) In the present context too (compare with Remarks 4.2.3 and 4.2.7), we should *not* make the mistake to believe that $\mathbb{P}_\mu^{\mathcal{M}}\big(\psi(x_1, \dots, x_n) \to \varphi(x_1, \dots, x_n)\big)$ is the same as $\mathbb{P}_\mu^{\mathcal{M}}\big(\varphi(x_1, \dots, x_n) \mid \psi(x_1, \dots, x_n)\big)$ in general. The reason is that if most $(a_1, \dots, a_n) \in M^n$ make $\varphi(x_1, \dots, x_n)$ false (in the structure $\mathcal{M}$), then the first probability is high and it is consistent with this situation that only a small proportion (possibly 0) of those $(a_1, \dots, a_n) \in M^n$ which satisfy $\psi(x_1, \dots, x_n)$ also satisfy $\varphi(x_1, \dots, x_n)$ and this means that the second probability is low.

We illustrate this with the example in Section 4.3.1. In particular we consider the structure called $\mathcal{A}$ in that section with domain A. (You need to go back to this example and recall the definitions there.) For any $a \in A$, $\mathcal{A} \not\models N(\hat{a}) \to T(\hat{a})$ if and only if $N(\hat{a})$ is true and $T(\hat{a})$ false. Only 1 of the 8 possibilities of $a \in A$ makes $N(\hat{a})$ true and $T(\hat{a})$ false. Hence $\mathbb{P}_\mu^{\mathcal{A}}(N(x) \to T(x)) = 7/8$. On the other hand we have

$$\mathbb{P}_\mu^{\mathcal{A}}\big(T(x) \mid N(x)\big) = \frac{\mathbb{P}_\mu^{\mathcal{A}}\big(T(x) \wedge N(x)\big)}{\mathbb{P}_\mu^{\mathcal{A}}\big(N(x)\big)} = \frac{0}{1/8} = 0.$$

Definition 4.3.7 Let $x_1, \dots, x_n$ be a sequence of variables and $\varphi(x_1, \dots, x_n)$ a formula such that all of its free variables belong to the sequence $x_1, \dots, x_n$ (but the formula need not have all of $x_1, \dots, x_n$ as free variables). By a little abuse of notation[6] we will also view $\varphi(x_1, \dots, x_n)$ as a binary random variable, with respect to the probability distribution $\mu^n : M^n \to \mathbb{R}$, as follows:

For every $(a_1, \dots, a_n) \in M^n$, define

$$\varphi(a_1, \dots, a_n) = \begin{cases} 1 & \text{if } \ \mathcal{M} \models \varphi(\hat{a}_1, \dots, \hat{a}_n) \\ 0 & \text{otherwise.} \end{cases}$$

Remark 4.3.8 Observe that $\varphi(x_1, \dots, x_n)$ viewed as a formula is *not* dependent on the structure $\mathcal{M}$, but $\varphi(x_1, \dots, x_n)$ viewed as a binary random variable *is* dependent on the structure $\mathcal{M}$, that is, if we change $\mathcal{M}$ then $\varphi(x_1, \dots, x_n)$ may become a different random variable.

Let us consider the formulas

$$\varphi(x_1, \dots, x_n),\ \psi(x_1, \dots, x_n),\ \theta_1(x_1, \dots, x_n), \dots, \theta_k(x_1, \dots, x_n)$$

as binary random variables. Then, from Definition 4.1.5, we know what it means for $\varphi(x_1, \dots, x_n)$ and $\psi(x_1, \dots, x_n)$ to be conditionally independent over

$$\theta_1(x_1, \dots, x_n), \dots, \theta_k(x_1, \dots, x_n).$$

[6]By abuse of notation I mean that the notation '$\varphi(x_1, \dots, x_n)$' has two meanings. It can denote a first-order formula, which is a syntactic expression, and it can denote a binary random variable, which is a function.

But nevertheless we rephrase the definition in the current context. We say that $\varphi(x_1, \ldots, x_n)$ and $\psi(x_1, \ldots, x_n)$ *are conditionally independent over*

$$\theta_1(x_1, \ldots, x_n), \ldots, \theta_k(x_1, \ldots, x_n).$$

if, for *every* choice of $j, i_1, \ldots, i_k \in \{0, 1\}$,

$$\begin{aligned} &\mu^n\big(\varphi(x_1, \ldots, x_n) = 1 \mid \psi(x_1, \ldots, x_n) = j, \\ &\qquad\qquad \theta_1(x_1, \ldots, x_n) = i_1, \ldots, \theta_k(x_1, \ldots, x_n) = i_k\big) \\ &= \mu^n\big(\varphi(x_1, \ldots, x_n) = 1 \mid \theta_1(x_1, \ldots, x_n) = i_1, \ldots, \theta_k(x_1, \ldots, x_n) = i_k\big) \end{aligned}$$

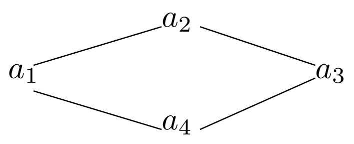
whenever both sidens are defined.

Remark 4.3.9 As mentioned directly after Definition 4.1.5, conditional independence over a set of random variables can also be characterized in terms of another identity which involves multiplication.

4.3.3 Example: applying the general definitions to a network

Consider the undirected graph in Figure 4.7. We can view it as a first-order structure,

Figure 4.7:



called $\mathcal{G}$ say, for a language with one binary relation symbol R in the following way: Let the domain of $\mathcal{G}$ be $G = \{a_1, a_2, a_3, a_4\}$ and let the interpretation of R be

$$R^{\mathcal{G}} = \{(a_1, a_2), (a_2, a_1), (a_1, a_3), (a_3, a_1), (a_2, a_4), (a_4, a_2), (a_3, a_4), (a_4, a_3)\}.$$

Let $\mu : G \to \mathbb{R}$ be the uniform probability distribution, that is, $\mu(a) = 1/4$ for every $a \in G$. Then $\mu^2(a, b) = 1/4^2 = 1/16$ for all $(a, b) \in G^2$ and

$$\mathbb{P}_\mu^{\mathcal{G}}(R(x, y)) = \mu^2\big(\{(a, b) \in G^2 : \mathcal{G} \models R(\hat{a}, \hat{b})\}\big) = 8 \cdot \frac{1}{16} = \frac{1}{2}.$$

and

$$\mathbb{P}_\mu^{\mathcal{G}}(R(x, y) \vee R(y, x)) = \mu^2\big(\{(a, b) \in G^2 : \mathcal{G} \models R(\hat{a}, \hat{b}) \vee R(\hat{b}, \hat{a})\}\big) = 8 \cdot \frac{1}{16} = \frac{1}{2}.$$

With words, the probability that for (not necessarily different) $a, b \in G$ there is an edge between a and b is $1/2$. But it may be more natural to consider randomly chosen *different* a and b from G. Then we need to condition on $x \neq y$. In other words, the probability that for two randomly chosen *different* a and b from G there is an undirected edge between a and b is

$$\mathbb{P}_\mu^{\mathcal{G}}(R(x, y) \mid x \neq y) = \frac{\mathbb{P}_\mu^{\mathcal{G}}(R(x, y) \wedge x \neq y)}{\mathbb{P}_\mu^{\mathcal{G}}(x \neq y)} = \frac{8 \cdot \frac{1}{16}}{\binom{4}{2}/16} = \frac{1/2}{6/16} = \frac{4}{3}.$$

Remark 4.3.10 Compare this example with the example of Section 4.2.6. The undirected graph is the same in the two examples. But the way of considering probabilities are entirely different in the two examples. In Section 4.2.6 we defined a set of structures (or possible worlds) $\mathbf{W}$ with domain $\{a_1, a_2, a_3, a_4\}$ and a probability distribution on $\mathbf{W}$. Then we could consider, for any first-order *sentence* φ, that is, first-order formula φ *without* free variables, the probability that φ is true in a structure in $\mathbf{W}$. In this example, on the other hand, we consider *only one* structure (or only one "world"), called $\mathcal{G}$. Moreover, the interesting case is to consider probabilities of formulas *with* free variables $\varphi(x_1, \ldots, x_n)$. The intuition is that $\mathbb{P}_{\mu}^{\mathcal{G}}(\varphi(x_1, \ldots, x_n))$ measures the likelihood that a randomly chosen "sample" $(a_1, \ldots, a_n) \in G^n$ will satisfy the formula $\varphi(x_1, \ldots, x_n)$ in the fixed structure (or world) $\mathcal{G}$. As hopefully indicated in the example in Section 4.3.1, the approach of considering probabilities according to Definition 4.3.1 is useful when we are given a set of objects as well as properties of these objects, or relations between them, and we want to compute the probability that a randomly chosen object from the data set has a certain property, or if we want to compute the probability that two, or more, randomly chosen objects from the data set are related in a certain way.

Chapter 5

Construction of a probabilistic model and inference

This chapter is an introduction to basic ideas and methods within the field of *statistical relational learning*, a subfield of artificial intelligence. The concepts treated in Chapter 4 will play a fundamental role. In statistical relational learning the basic problem can roughly be described as follows: We start with a set of data, where the data consists of a set of objects and relations between the objects. Such a set of data is often called the "learning set/sample", the "set of examples", or something similar. A relation may involve two or more objects which is why the expression "multidimensional data" is sometimes used. For example, the relation "x and y are friends" could be called 2-dimensional, but we will call such a relation 2-ary since we stick to terminology from first-order logic. A 1-ary (or 1-dimensional) relation, for example "x is a student", could also be called a "property" (that an object may or may not have); in the context of databases, properties are often called "attributes". To express relations and properties we will use first-order logic and this means that the "learning set" will be represented by a first-order structure. But other declarative languages, such as SQL or Prolog, can also be used. Given the data (the set of objects and the relations between them) the task is to find a suitable probabilistic model that describes the data. The probabilistic models considered in statistical relational learning are usually so-called "graphical models" which use graphs to describe dependencies and (conditional) independencies between random variables. The choice of graphical model in this text is the Bayesian network. The final step is to use the Bayesian network to make predictions on other sets of objects than the "learning set". If want to predict the probability of some objects satisfying a relation that occurs (in the form of a binary random variable) in the Bayesian network, then this is straightforward. But if we want to predict (or estimate) the probability of an event which is described by a more complex first-order formula, then we need to use the "probabilities on possible worlds approach" discussed in Section 4.2 to be on firm mathematical ground.

Section 5.1 describes, by means of a general algorithm, how one can derive a Bayesian network from a first-order structure (which represents the "learning set") with a probability distribution on its domain. The "probabilities on domains approach" to logic and probability which was discussed in Section 4.3 is used in the context of this algorithm. Section 5.2 illustrates by means of examples how to construct a Bayesian network with the algorithm from Section 5.1 and how to use the Bayesian network for prediction. In Section 5.3 we consider the situation when the "learning set" has incomplete/partial information. In order to construct a lifted Bayesian network which fits the learning sample as well as possible we need to estimate probabilities and conditional probabilities of the random variables of interest. By means of an example I will illustrate how the so-called *EM algorithm* works to find such estimates.

5.1 Construction of a Bayesian network from a first-order structure

For all of this section we let L be a first-order language and $\mathcal{M}$ an L-structure with domain M. We assume that for every $a \in M$ there is a constant symbol $\hat{a}$ which is interpreted as a (or with symbols, such that $\hat{a}^{\mathcal{M}} = a$). Furthermore, we let $\mu : M \to \mathbb{R}$ be a probability distribution. (For example, if M is finite, then μ may be the uniform probability measure which gives every $a \in M$ the same probability.) Recall from Section 4.1 μ induces a probability distribution $\mu^n : M^n \to \mathbb{R}$ for every natural number $n > 0$ (and we indentify M^1 and μ^1 with M and μ, respectively). Let $\varphi_1(x_1, \ldots, x_n), \ldots, \varphi_m(x_1, \ldots, x_n)$ be L-formulas such that all free variables in each φ_i belong to the sequence $x_1, \ldots, x_n$ (but we do not require that every x_i occur as a free variable in every φ_j). We will view these formulas as binary random variables according to Definition 4.3.7.

The intuition is that the structure $\mathcal{M}$ is our "training example", in other words, $\mathcal{M}$ can be viewed as a formalization of a "training set/example" of objects/individuals and properties that they have or relationships between them. From the training example $\mathcal{M}$ we want to derive a lifted Bayesian network which exhibits the (conditional) dependencies and independencies between the binary random variables

$$\varphi_1(x_1, \ldots, x_n), \ldots, \varphi_m(x_1, \ldots, x_n)$$

and we also want to compute the conditional probabilities which are associated with the vertices of the *directed acyclic graph* (DAG) of the lifted Bayesian network. Each binary random variable $\varphi_i(x_1, \ldots, x_n)$, $i = 1, \ldots, m$, is a vertex of the DAG that is to be constructed. To simplify notation we will abbreviate '$\varphi_i(x_1, \ldots, x_n)$' by '$\varphi_i$'.

Recall the discussion about conditional independence over a set (or sequence) of binary random variables in Section 4.3.2.

Algorithm[1] for finding the arrows of a DAG corresponding to a lifted Bayesian network:

```
Set $k := 1$ and as long as $k < m$ do:

    For every $i = 1, \ldots, k$, add an arrow from $\varphi_i$ to $\varphi_{k+1}$.
    Set $I := \{1, \ldots, k\}$.
    For $i = 1, \ldots, k$ do:

        If $\varphi_{k+1}$ and $\varphi_i$ are conditionally independent over $(\varphi_j : j \in I \setminus \{i\})$, then remove
        $i$ from $I$ (i.e. set $I := I \setminus \{i\}$) and remove the arrow from $\varphi_i$ to $\varphi_{k+1}$. Otherwise
        do nothing.

    Set $k := k + 1$.
```

Next, for every vertex φ_k with parents $\varphi_{i_1}, \ldots, \varphi_{i_s}$, say, one computes the conditional probability

$$\mu^n\big(\varphi_k = 1 \mid \varphi_{i_1} = b_1, \ldots, \varphi_{i_s} = b_s\big) \text{ for every choice of } b_1, \ldots, b_s \in \{0, 1\}.$$

[1]This is one version of the so-called PC-algorithm.

If φ_k has no parent then one just computes $\mu^n(\varphi_k)$. Then the joint probability distribution on $(\varphi_1, \ldots, \varphi_m)$ induced by μ^n is determined by the conditional probabilities associated to the vertices of the DAG. More precisely, for every choice of $b_1, \ldots, b_m \in \{0,1\}$ the probability that $\varphi_i = b_i$ for all $i = 1, \ldots, m$ can be computed by only using the conditional probabilities associated with vertices of the DAG. One can also verify that the other conditions in the definition of a Bayesian network (Definition 4.1.6) are satisfied, so the DAG constructed by the algorithm together with the conditional probabilities associated with its vertices is indeed a Bayesian network. Thus the DAG together with the associated conditional probabilities (to each vertex) is a Bayesian network for the binary random variables and $\varphi_1, \ldots, \varphi_m$ and the probability distribution $\mu^n : M^n \to \mathbb{R}$.

The reader may have various questions regarding the construction of the Bayesian network above. Hopefully some of them are answered by the remarks that follow.

Remark 5.1.1 (Atomic formulas) If the structure $\mathcal{M}$ is a mathematical representation of a database, or of some other "training sample"[2], then it is natural to represent relations of interest by relation symbols and the atomic formulas obtained from these. With all assumptions made above, if we add the assumption that $\varphi_i(x_1, \ldots, x_n)$ is atomic, then it means that for some relation symbol, say R of arity r, we have $r \leq n$ and $\varphi_i(x_1, \ldots, x_n)$ has the form $R(x_{j_1}, \ldots, x_{j_r})$ where $x_{j_1}, \ldots, x_{j_r} \in \{x_1, \ldots, x_n\}$.

Remark 5.1.2 (Approximate independencies may be used in practice) Suppose that the structure $\mathcal{M}$ is a mathematical representation of a database, or of some other "training example". Then it is unlikely that any two random variables are conditionally independent over some set of random variables *in the exact sense of our definitions.* In other words, it is unlikely that the following identity will hold exactly for any nontrivial choice of $i, j, k_1, \ldots, k_l$ and $a, b_1, \ldots, b_l \in \{0,1\}$:

$$\mu^n\big(\varphi_i = 1 \mid \varphi_j = a, \varphi_{k_1} = b_1, \ldots, \varphi_{k_l} = b_l\big) = \mu^n\big(\varphi_i = 1 \mid \varphi_{k_1} = b_1, \ldots, \varphi_{k_l} = b_l\big).$$

But there may be some small $\varepsilon > 0$ such that if, for every choice of $b_1, \ldots, b_l$, the difference between the left and right hand sides in the above equation is at most ε. Then we may consider, "for practical purposes", φ_i and φ_j to be independent over $\varphi_{k_1}, \ldots, \varphi_{k_l}$. What an appropriate "error marginal" φ might be depends, of course, on the particular application.

Remark 5.1.3 (The problem of "overfitting") In applications of predictive models such as Bayesian networks to the "real world" one faces the problem of *overfitting*, because the probabilities of random variables on a training example are unlikely to be exactly the same as in the whole population. Roughly speaking, a (lifted) Bayesian network *overfits* the training example if it describes the probability distribution on the training example so well that the probabilities predicted by the Bayesian network are not representative of the whole population (the whole domain of interest). Another aspect of overfitting is that one usually seeks a probabilistic model (e.g. Bayesian network) which is not too complex, because very complex models (e.g. very connected Bayesian networks) are harder to understand and more time consuming to use in practice. Although avoiding overfitting is an important issue within machine learning this text will not consider this issue. But a general idea when trying to avoid overfittning is to remove arrows in the Bayesian network which correspond to "weak" probabilistic dependencies, where it is up to the model constructor to determine what (sufficently) "weak" means. This is related to the discussion in Remark 5.1.2 about the "threshold" ε for considering two random variables to be independent over a set of other random variables.

[2]As for example in Section 4.3.1.

Remark 5.1.4 (The relevance of the ordering of $\varphi_1, \ldots, \varphi_m$) In the algorithm above for constructing a DAG we treated the random variables according to the given order $\varphi_1, \ldots, \varphi_m$. In general, if we change the ordering, for example to $\varphi_m, \ldots, \varphi_1$, we may get a *different* DAG. In other words the constructed DAG depends on the ordering of the random variables $\varphi_1, \ldots, \varphi_m$ (and there are $m!$ orderings of them). A DAG obtained from one ordering may be more informative regarding conditional independencies and dependencies than a DAG obtained from another ordering. (See the example in Section 5.2.1 below.) The field of (probabilistic) graphical models (see e.g. [18]) studies the topic of finding an "optimal" DAG in more detail.

Remark 5.1.5 (Computational complexity) Suppose that the structure $\mathcal{M}$ has finite domain M, so $|M|$, the size (or more precisely cardinality) of M, is a finite number. If $\mathcal{M}$ represents a database (or some other "training example") then one expects $|M|$ to be much larger than the number of binary random variables $\varphi_1, \ldots, \varphi_m$ which is m. Let us assume that m is constant, while $|M|$ may grow, that is we may replace $\mathcal{M}$ with a similar structure but with larger domain. It follows that the number of orderings of $\varphi_1, \ldots, \varphi_m$ and the number of possible choices of $b_1, \ldots, b_m \in \{0, 1\}$ is constant. Let us furthermore assume that all $\varphi_i(x_1, \ldots, x_n)$ are atomic formulas (when viewed as formulas). And finally let us assume that $\mu : M \to \mathbb{R}$ is uniform, that is, $\mu(a) = 1/|M|$ for every $a \in M$. Then, for every formula $\psi(x_1, \ldots, x_n)$,

$$\mu^n(\psi(x_1, \ldots, x_n)) = \frac{\mu^n\big(\{(a_1, \ldots, a_n) \in M^n : \mathcal{M} \models \psi(\hat{a}_1, \ldots, \hat{a}_n)\}\big)}{|M|^n}$$

so $\mu^n(\psi(x_1, \ldots, x_n))$ is determined by the number of $(a_1, \ldots, a_n) \in M^n$ such that $\mathcal{M} \models \psi(\hat{a}_1, \ldots, \hat{a}_n)$. Assuming that there is a constant that gives an upper bound for the time of checking, for any *atomic* formulas $\psi(x_1, \ldots, x_n)$ and any $(a_1, \ldots, a_n) \in M^n$ whether $\mathcal{M} \models \psi(\hat{a}_1, \ldots, \hat{a}_n)$ or not, it follows that the time complexity of the algorithm is $c|M|^n$ for some constant c. In other words, the number of "basic steps" that the algorithm needs to carry out before it has constructed the Bayesian network is $c|M|^n$ for some constant c (and under the assumptions made).

Remark 5.1.6 (About "lifting" the network to other domains) Suppose that we use the Bayesian network constructed above to make predictions on some other domain, say D, via the "probabilities on possible worlds approach", as discussed in Sections 4.2.9 and 4.2.10. Then each random variable $\varphi_i(x_1, \ldots, x_n)$ on the probability space M^n will give rise to a multitude of random variables on the probability space $\mathbf{W}_D$ consisting of all L-structures with domain D, namely a random variable $\varphi_i(d_1, \ldots, d_n) : \mathbf{W}_D \to \{01\}$ for every $(d_1, \ldots, d_n) \in D^n$. It is reasonable to require that a probability distribution $\mathbb{P}_D$ on $\mathbf{W}_D$ that "respects" the Bayesian network (constructed from $\mathcal{M}$ and μ) should give rise to the same conditional (in)dependencies and conditional probabilities as the Bayesian network describes. If all $\varphi_i(x_1, \ldots, x_n)$ correspond to atomic L-formulas, then such a probability distribution $\mathbb{P}_D$ (on $\mathbf{W}_D$) can be found. This was illustrated in Section 4.2.9 and will be illustrated again in sections 5.2.1 and 5.2.2. In (the more theoretical) Section 6.2 a kind of fairly general type of Bayesian network is defined and it is also defined exactly how it gives rise to a probability distribution on $\mathbf{W}_D$ where D is any finite set.

When a Bayesian network is constructed from one structure (representing a set of data), as done above, and is then used to derive a probability distribution on the set of structures on some other domain (representing the possible states the world/system can have) the the Bayesian network is sometimes called a *lifted*, *parametrized* or *template based* Bayesian network.

5.2 Examples of model construction and prediction

In this section we will see consider a few examples in which we are given a "learning sample" of data. Our first task is to choose a suitable first-order language L and represent the learning sample as an L-structure, say $\mathcal{A}$, and then, for some selected L-formulas, viewed as binary random variables, to find a lifted Bayesian network which describes the joint probability distribution on these formulas. In this step we use the algorithm from Section 5.1. This involves using a probability distribution on the domain of $\mathcal{A}$.

The next step is to use the Bayesian network (our probabilistic model) to make predictions of events being true in other domains than the objects of the learning sample. In this step we use the "probabilities on possible worlds approach" from Section 4.2. Somewhat more precisely, suppose that we want to use our Bayesian network to predict probabilities of events on a set of objects D. Then we can define a suitable set $\mathbf{W}$ of L-structures with domain D and then, using the Bayesian network, we define a probability distribution $\mu : \mathbf{W} \to \mathbb{R}$ which "fits" the Bayesian network. We already carried out this procedure in the example in Section 4.2.9. So in this section we repeat the same sort of reasoning but to other examples. The probability distribution $\mu : \mathbf{W} \to \mathbb{R}$ gives rise to $\mathbb{P}_\mu$ as defined in Section 4.2.10 and we can, in principle[3], use $\mathbb{P}_\mu$ to predict the probability of any event that can be expressed by an L-sentence.

Remark: The examples below are of course unrealistic in the sense that the training/learning sets are very small because I have chosen the examples so that (conditional) probabilities are relatively easy to compute by hand and so that we get certain (conditional) independencies. Moreover, we ignore the issue of overfitting discussed in Remark 5.1.3. The point with the examples is to illustrate some general ideas and principles which, in "real world" applications, need to be complemented by other techniques which are beyond the scope of this text.

5.2.1 Example: relating consumption of various items

Suppose that we have a database containing information about the items that customers has bought. In our simplification we consider only three items, called I_1, I_2 and I_3, and the database has information about 12 customers, called $A, B, C, \dots, L$. The database is illustrated by the table in Figure 5.1 below where 'yes' means that the customer buys the item and (of course) 'no' means that the customer does not buy it.

Our first task is to represent the database mathematically as a first-order structure. Similarly as in Section 4.3.1 this can be done in more than one way. Here we take the simple and unsophisticated approach to represent each item by a relation symbol. In fact let us consider I_1, I_2 and I_3 as relation symbols and let $I_1(x)$ represent the statement "x bought item I_1" and similarly for I_2 and I_3. The first-order language that we consider has only these three relation symbols. Let $\mathcal{M}$ be the (first-order) structure with domain

$$M = \{A, B, C, \dots, L\}$$

and such that

$$\begin{aligned} I_1^{\mathcal{M}} &= \{A, D, F, G, J, L\}, \\ I_2^{\mathcal{M}} &= \{B, D, G, I, J, L\}, \\ I_3^{\mathcal{M}} &= \{A, C, D, F, G, I, K, L\}. \end{aligned}$$

[3] I say "in principle" because the computations can in general be quite difficult.

Figure 5.1:

customer	buys I_1	buys I_2	buys I_3
A	yes	no	yes
B	no	yes	no
C	no	no	yes
D	yes	yes	yes
E	no	no	no
F	yes	no	yes
G	yes	yes	yes
H	no	no	no
I	no	yes	yes
J	yes	yes	yes
K	no	no	yes
L	yes	yes	yes

Then $\mathcal{M}$ is a first-order structure which represents the information of the table above. Let μ be the uniform probability distribution on M, so $\mu(a) = 1/|M| = 1/12$ for every $a \in M$. As explained in Definition 4.3.7 we can view the formulas $I_1(x)$, $I_2(x)$ and $I_3(x)$ as binary random variables.

Our next step is to find a lifted Bayesian network which describes the (in)dependencies and conditional probabilities between the binary random variables $I_1(x)$, $I_2(x)$ and $I_3(x)$. We use the algorithm from Section 5.1 and first we consider the random variables ordered as: $I_1(x)$, $I_2(x)$, $I_3(x)$. When the algorithm carries out the main loop for $k = 1$ it finds that $I_1(x)$ and $I_2(x)$ are dependent, because $\mu(I_1 = 1) = 1/2$, $\mu(I_2 = 1 \mid I_1 = 1) = 2/3$ and $\mu(I_2 = 1 \mid I_1 = 0) = 1/3$. Hence there will be an arrow from $I_1(x)$ to $I_2(x)$. When the algorithm carries out the main loop for $k = 2$ it finds that I_3 is conditionally independent from I_2 over I_1 and that I_3 is dependent on I_1, because

$$\begin{aligned}
&\mu(I_3 = 1 \mid I_2 = 0, I_1 = 0) = 1/2 = \mu(I_3 = 1 \mid I_1 = 0),\\
&\mu(I_3 = 1 \mid I_2 = 0, I_1 = 1) = 1 = \mu(I_3 = 1 \mid I_1 = 1),\\
&\mu(I_3 = 1 \mid I_2 = 1, I_1 = 0) = 1/2 = \mu(I_3 = 1 \mid I_1 = 0),\\
&\mu(I_3 = 1 \mid I_2 = 1, I_1 = 1) = 1 = \mu(I_3 = 1 \mid I_1 = 1),\\
&\mu(I_3 = 1 \mid I_1 = 1) = 1 \text{ and}\\
&\mu(I_3 = 1) = 3/4.
\end{aligned}$$

Hence there will be an arrow from $I_1(x)$ to $I_3(x)$ but no arrow from $I_2(x)$ to $I_3(x)$. So the lifted Bayesian network is like in Figure 5.2 where we have written '$\mu(I_1(x)) = 1/2$' instead of '$\mu(I_1 = 1)$' or '$\mu(I_1(x) = 1)$' and similarly for the other (conditional) probabilities in the table.

If instead we would apply the same algorithm to the order I_3, I_1, I_2, then we get the lifted Bayesian network in Figure 5.3 where we note that $\mu(I_2(x) \mid \neg I_3(x) \wedge I_1(x))$ is undefined as $\mu(\neg I_3(x) \wedge I_1(x)) = 0$.

I would argue that the lifted Bayesian network in Figure 5.2 is more informative than that in Figure 5.3, because the DAG in Figure 5.2 has fewer arrows and hence it reveals more conditional independencies (well just one in this case). It is dependencies between random variables that make predictions harder. So in the rest of this example we will

Figure 5.2:

$$\begin{tikzcd}
 & I_1(x) \arrow[dl] \arrow[dr] & \\
I_2(x) & & I_3(x)
\end{tikzcd}$$

$\mu(I_1(x)) = 1/2$	$\mu(I_2(x) \mid I_1(x)) = 2/3$	$\mu(I_3(x) \mid I_1(x)) = 1$
	$\mu(I_2(x) \mid \neg I_1(x)) = 1/3$	$\mu(I_3(x) \mid \neg I_1(x)) = 1/2$

Figure 5.3:

$$\begin{tikzcd}
 & I_3(x) \arrow[dl] \arrow[dr] & \\
I_1(x) \arrow[rr] & & I_2(x)
\end{tikzcd}$$

$\mu(I_3(x)) = 3/4$	$\mu(I_1(x) \mid I_3(x)) = 2/3$	$\mu(I_2(x) \mid I_3(x) \wedge I_1(x)) = 2/3$
	$\mu(I_1(x) \mid \neg I_3(x)) = 0$	$\mu(I_2(x) \mid I_3(x) \wedge \neg I_1(x)) = 1/3$
		$\mu(I_2(x) \mid \neg I_3(x) \wedge I_1(x))$ undefined
		$\mu(I_2(x) \mid \neg I_3(x) \wedge \neg I_1(x)) = 1/3$

consider the lifted Bayesian network in Figure 5.2.

Recall that the lifted Bayesian network was constructed from the structure called $\mathcal{M}$ above and that we considered a probability distribution on the domain M of $\mathcal{M}$, according to the approach in Section 4.3. Now that we have our lifted Bayesian network we may want to use it to predict how customers from a given population of individuals will consume items (among I_1, I_2 and I_3). Let N denote the set (or population) of individuals for which we which to predict consumption behaviour according to our model, that is, our lifted Bayesian network in Figure 5.2.

We would like to be able to predict the probability that a statement, formalized by a first-order formula $\psi(x_1, \ldots, x_n)$, holds if the variables $x_1, \ldots, x_n$ refer to randomly chosen individuals from N. In particular we may be interested in the probability that a sentence (i.e. formula without free variables) θ holds in the population N. If $\psi(x)$ is a formula of the form $\varphi_1(x) \wedge \varphi_2(x) \wedge \varphi_3(x)$ where $\varphi_i(x)$ is $I_i(x)$ or $\neg I_i(x)$ for each $i = 1, 2, 3$, then we can predict, for any the probability that $\psi(x)$ is satisfied by a randomly chosen individual from N by using the lifted Bayesian network alone. But if we want to predict the probability that a more complicated formula, such as for example the sentence

$$\forall x\big((I_1(x) \wedge I_2(x)) \to I_3(x)\big) \quad \text{or} \quad \forall x\big(I_3(x) \to (I_1(x) \vee I_2(x))\big)$$

is true for the population N, then it is not clear (I think) how to use the lifted Bayesian network alone, without some added conceptual framework. To deal with this issue in a mathematically precise way we need to define precisely what we mean if we say that a first-order sentence has a certain probability of being true in the population N. A way of getting this precision is to use the "possible worlds" approach from Section 4.2 to build a mathematical model of the situation. This means that we consider a set $\mathbf{W}$ of first-order structures – or "possible worlds" or "possible states" – where each first-order structure in $\mathbf{W}$ is a mathematical representation of a possible "state" that the population N could be in. We do not know which items an individual from N will buy; if we knew it there would be no need for prediction. Each state (or "possible world") specifies for every individual in N which items that individual buys. Some states may however be more likely than others which is why we need a probability distribution on $\mathbf{W}$. To define the probability distribution $\lambda : \mathbf{W} \to \mathbb{R}$ we use the lifted Bayesian network in Figure 5.2.

We proceed similarly as in the example in Section 4.2.9. First, to simplify notation let us assume that $N = \{1, 2, \dots, n\}$. We let $\mathbf{W}$ be the set of all first-order structures with domain N for the first-order language with unary relation symbols I_1, I_2, I_3 and a constant symbol $\hat{i}$ for each $i \in N$ where $\hat{i}$ is always interpreted as i. According to the lifted Bayesian network in Figure 5.2, for every $i \in N$, we stipulate that the formula in the left kolumn in Figure 5.4 holds with the probability given in the right column (on the same row) independently of what is the case for $j \neq i$.

Figure 5.4:

$I_1(\hat{i}) \wedge I_2(\hat{i}) \wedge I_3(\hat{i})$	$\frac{1}{2} \cdot \frac{2}{3} \cdot 1 = 1/3$
$\neg I_1(\hat{i}) \wedge I_2(\hat{i}) \wedge I_3(\hat{i})$	$\frac{1}{2} \cdot \frac{1}{3} \cdot \frac{1}{2} = 1/12$
$I_1(\hat{i}) \wedge \neg I_2(\hat{i}) \wedge I_3(\hat{i})$	$\frac{1}{2} \cdot \frac{1}{3} \cdot 1 = 1/6$
$I_1(\hat{i}) \wedge I_2(\hat{i}) \wedge \neg I_3(\hat{i})$	$\frac{1}{2} \cdot \frac{2}{3} \cdot 0 = 0$
$\neg I_1(\hat{i}) \wedge \neg I_2(\hat{i}) \wedge I_3(\hat{i})$	$\frac{1}{2} \cdot \frac{2}{3} \cdot \frac{1}{2} = 1/6$
$I_1(\hat{i}) \wedge \neg I_2(\hat{i}) \wedge \neg I_3(\hat{i})$	$\frac{1}{2} \cdot \frac{1}{3} \cdot 0 = 0$
$\neg I_1(\hat{i}) \wedge I_2(\hat{i}) \wedge \neg I_3(\hat{i})$	$\frac{1}{2} \cdot \frac{1}{3} \cdot \frac{1}{2} = 1/12$
$\neg I_1(\hat{i}) \wedge \neg I_2(\hat{i}) \wedge \neg I_3(\hat{i})$	$\frac{1}{2} \cdot \frac{2}{3} \cdot \frac{1}{2} = 1/6$

Let $\mathcal{N} \in \mathbf{W}$ and, for $l = 1, \dots, 8$, let k_l be the number of $i \in N$ such that the formula on the l^{th} row in the table in Figure 5.4 holds in $\mathcal{N}$. Then we define (where we recall the convention that $0^0 = 1$ but $0^c = 0$ whenever $c > 0$)

$$\lambda(\mathcal{N}) = (1/3)^{k_1}(1/12)^{k_2}(1/6)^{k_3}0^{k_4}(1/6)^{k_5}0^{k_6}(1/12)^{k_7}(1/6)^{k_8}.$$

Note that if $k_4 > 0$ or $k_6 > 0$ then $\lambda(\mathcal{N}) = 0$. As usual we extend λ to a function on the set of all subsets of $\mathbf{W}$ as follows:

$$\text{For every } \mathbf{X} \subseteq \mathbf{W}, \lambda(\mathbf{X}) = \sum_{\mathcal{N} \in \mathbf{X}} \lambda(\mathcal{N}).$$

Then (also as usual) we define, for every sentence φ,

$$\mathbb{P}_\lambda(\varphi) = \lambda\big(\{\mathcal{N} \in \mathbf{W} : \mathcal{N} \models \varphi\}\big).$$

Now it is possible to evaluate (for example)

$$\mathbb{P}_\lambda\big[\forall x\big((I_1(x) \wedge I_2(x)) \to I_3(x)\big)\big].$$

In this case it is convenient to notice that $\forall x\big((I_1(x) \wedge I_2(x)) \to I_3(x)\big)$ is logically equivalent to

$$\forall x\big(\neg I_1(x) \vee \neg I_2(x) \vee I_3(x)\big).$$

The *negation* of it is equivalent to $\exists x\big(I_1(x) \wedge I_2(x) \wedge \neg I_3(x)\big)$ and we se from the definitions of λ and $\mathbb{P}_\lambda$ and the table in Figure 5.4 that if $\mathcal{N} \models \exists x\big(I_1(x) \wedge I_2(x) \wedge \neg I_3(x)\big)$ then $\lambda(\mathcal{N}) = 0$, so

$$\mathbb{P}_\lambda\big[\exists x\big(I_1(x) \wedge I_2(x) \wedge \neg I_3(x)\big)\big] = 0.$$

It follows (from Proposition 4.2.5) that $\mathbb{P}_\lambda\big[\forall x\big((I_1(x) \wedge I_2(x)) \to I_3(x)\big)\big] = 1$.

Now let us consider the formula $\forall x\big(I_3(x) \to (I_1(x) \vee I_2(x))\big)$. It is equivalent to $\forall x\big(I_1(x) \vee I_2(x) \vee \neg I_3(x)\big)$. It follows that it is false in $\mathcal{N} \in \mathbf{W}$ if and only if there is at least one $i \in N$ such that $\mathcal{N} \models \neg I_1(\hat{\imath}) \wedge \neg I_2(\hat{\imath}) \wedge I_3(\hat{\imath})$. For each $i \in N$ the probability (see the table in Figure 5.4) that that $\mathcal{N} \not\models \neg I_1(\hat{\imath}) \wedge \neg I_2(\hat{\imath}) \wedge I_3(\hat{\imath})$ is $5/6$. So the probability that a randomly chosen $\mathcal{N}$ satisfies $\forall x\big(I_1(x) \vee I_2(x) \vee \neg I_3(x)\big)$ is $(5/6)^n$. In other words,

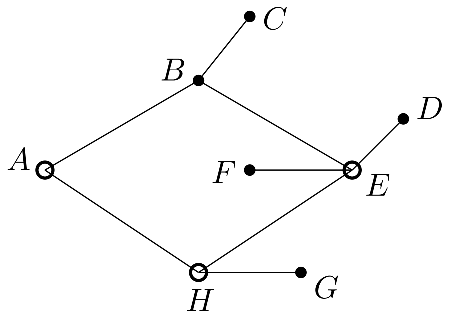
$$\mathbb{P}_\lambda\big[\forall x\big(I_3(x) \to (I_1(x) \vee I_2(x))\big)\big] = (5/6)^n$$

where we may note that this probability tends to 0 as n tends to infinity.

5.2.2 Example: small and large cities and airline connections

Suppose that we have eight cities which we denote A, B, C, D, E, F, G, H. Some of these cities are (according to some measure) considered big and others not. Between some of the cities there is a direct airline connection and between others not. Figure 5.5 illustrates the situation, where a big city is represented by a circle and a small city by a dot. Direct connections are illustrated by edges.

Figure 5.5:



As in Section 5.2.1 our first goal is to construct a lifted Bayesian network from the "training example" in Figure 5.5 as described in Section 5.1. We first formalize the situation with a first-order structure. We will call our structure $\mathcal{N}$ and its domain will be $N = \{A, B, C, D, E, F, G, H\}$. Let '*Big*' be a unary relation symbol and '*Direct*' a binary relation symbol. We think of $Big(x)$ as expressing that "x is big" and $Direct(x, y)$ as expressing that "there is a direct connection between x and y". According to Figure 5.5 the interpretation of *Big* should be $Big^{\mathcal{N}} = \{A, E, H\}$ and the interpretation of *Direct* should be

$$\begin{aligned} Direct^{\mathcal{N}} = \{&(A, B), (B, A), (A, H), (H, A), (B, C), (C, A), (B, E), (E, B), \\ &(E, D), (D, E), (E, F), (F, A), (E, H), (H, E), (H, G), (G, H)\}. \end{aligned}$$

The lifted Bayesian network that we construct will have the binary random variables $Big(x)$, $Big(y)$ and $Direct(x, y)$. Let $\mu : N \to \mathbb{N}$ be the uniform probability distribution, so $\mu(a) = 1/8$ for all $a \in N$. Since the highest arity these formulas, also seen as random variables, is 2 (since $Direct(x, y)$ has arity 2 and $Big(x)$ and $Big(y)$ have arity 1) we work, technically speaking, with the set N^2 and the probability distribution $\mu^2 : N^2 \to \mathbb{N}$ (where $\mu(a, b) = \mu(a) \cdot \mu(b) = (1/8)^2 = 1/64$ for all $(a, b) \in N^2$). Thus we will consider $Big(x)$ as having a "dummy" variable y and $Big(y)$ as having a "dummy" variable x. Alternatively we can replace $Big(x)$ by '$Big(x) \wedge y = y$' and $Big(y)$ by '$x = x \wedge Big(y)$'. By Lemma 4.3.4, $\mathbb{P}^{\mathcal{N}}_\mu(Big(x)) = \mathbb{P}^{\mathcal{N}}_\mu(Big(x)) \wedge y = y)$ and $\mathbb{P}^{\mathcal{N}}_\mu(Big(y)) = \mathbb{P}^{\mathcal{N}}_\mu(x = x \wedge Big(y))$, so the probabilities are not affected by whether we view $Big(x)$ as a formula in (only) the free variable x or in the free variables x and y; and dually, the probabilities are not affected by whether we view $Big(x)$ as a binary random variable on N or on N^2.

If we consider the random variables in the order $Big(x), Big(y)$ and $Direct(x, y)$ and follow the algorithm in Section 5.1, then we get the lifted Bayesian network in Figure 5.6.

Figure 5.6:

$$\begin{tikzcd}
Big(x) \arrow[dr] & & Big(y) \arrow[dl] \\
& Direct(x,y) &
\end{tikzcd}$$

$\mu(Big(x)) = \mu(Big(y)) = 3/8$

$\mu(Direct(x, y) \mid x \neq y \wedge Big(x) \wedge Big(y)) = 2/3$
$\mu(Direct(x, y) \mid x \neq y \wedge Big(x) \wedge \neg Big(y)) = 4/15$
$\mu(Direct(x, y) \mid x \neq y \wedge \neg Big(x) \wedge Big(y)) = 4/15$
$\mu(Direct(x, y) \mid x \neq y \wedge \neg Big(x) \wedge \neg Big(y)) = 1/10$

$\mu(Direct(x, y) \mid x = y) = 0$

Observe that although $x = y$ is not a random variable of the lifted bayesian network in Figure 5.6 we have nevertheless conditioned the probability of $Direct(x, y)$ on $x \neq y$ and $x = y$. The reason is that we want to make it clear that (according to Figure 5.5) there is no direct flight from any city to itself. One could also consider an approach where $x = y$ (or $x \neq y$) is added to the random variables and included in the lifted Bayesian network. However, in a small domain, as in Figure 5.5, we get a rather strong dependency between $x = y$ on the one hand and $Big(x)$ and $Big(y)$ on the other hand. We have $\mathbb{P}^{\mathcal{N}}_\mu(x = y) = 1/8$ while $\mathbb{P}^{\mathcal{N}}_\mu(x = y \mid Big(x) \wedge Big(y)) = 1/3$. If we had a much larger domain in which about 3/8 of the cities where big, then the dependency between the random variable $x = y$ and the random variables $Big(x)$ and $Big(y)$ would be much weaker and as the size of the domain tends to infinity the dependency between $x = y$ and $Big(x)$ and $Big(y)$ fades away. Since we eventually want to predict the behaviour on large domains it would be more appropriate (if we added $x = y$ to the random variables of the lifted Bayesian network) to treat $x = y$ as being independent from both $Big(x)$ and $Big(y)$ *although* in the small "training example" in Figure 5.5 this is *not* the case.

Unlike the case of the formula $R(x, y)$ in Section 4.2.9 we interpret $Direct(x, y)$ as being a symmetric relation, in other words if $Direct(x, y)$ holds then $Direct(y, x)$ holds. We could have made this information explicit in the network by adding the random variable $Direct(y, x)$ to the network and adding an arrow from $Direct(x, y)$ to $Direct(y, x)$ with the following associated conditional probabilities:

$$\mu(Direct(y,x) \mid Direct(x,y)) = 1,$$
$$\mu(Direct(y,x) \mid \neg Direct(x,y)) = 0.$$

Our next step is to construct a mathematical model for prediction; for example we might want to predict the probability that there is a direct flight between a randomly chosen big city and a randomly chosen small city; or we may want to predict the probability that for two randomly chosen small cities one can fly from one to the other with at most one stop; or we may want to predict the probability that for every pair of different small cities it is possible to fly from one to the other with at most one stop. The difference between the two last statements is more clear when formulated in first-order logic. Our predictions are based on the "training example" in Figure 5.5, so if the training example is not representative of the rest of the world, then the predictions will be inaccurate; and if the training example is sufficiently representative of the rest of the world, then the predictions will be more accurate. As in the example in Section 5.2.1, the mathematical model of prediction that we will use is based on the "possible worlds approach" wich was discussed in Section 4.2. Compare in particular with the example in Section 4.2.9 where we deal with a similar lifted Bayesian network.

Let $D = \{1, \dots, n\}$ be a domain. (It is irrelevant what the members of D are called so we just label them as $1, \dots, n$ where n is the number of elements in the domain.) We let L denote the first-order language with the relation symbols *Big* and *Direct* and a constant symbol $\hat{\imath}$ for every $i \in D$. Let $\mathbf{W}$ be the set of all L-structures $\mathcal{A}$ with domain D and such that *Direct* is interpreted as a symmetric relation (that is, such that $\mathcal{A} \models \forall x \forall y(Direct(x,y) \rightarrow Direct(y,x))$) and $\hat{\imath}$ is interpreted as i for every $i \in D$.

We will now define a probability distribution $\lambda : \mathbf{W} \rightarrow \mathbb{R}$. The motivation for how λ is defined is analogous to the motivation for how the distribution called μ is defined in Section 4.2.9.

For every $\mathcal{A} \in \mathbf{W}$, if

$q =$ the number of $i \in D$ such that $\mathcal{A} \models Big(\hat{\imath})$,

$r_1 =$ half of the number of (ordered) pairs $(i,j) \in D^2$ such that $i \neq j$ and $\mathcal{A} \models Direct(\hat{\imath},\hat{\jmath}) \wedge Big(\hat{\imath}) \wedge Big(\hat{\jmath})$,

$r_2 =$ half of the number of (ordered) pairs $(i,j) \in D^2$ such that $i \neq j$ and $\mathcal{A} \models Direct(\hat{\imath},\hat{\jmath}) \wedge Big(\hat{\imath}) \wedge \neg Big(\hat{\jmath})$,

$r_3 =$ half of the number of (ordered) pairs $(i,j) \in D^2$ such that $i \neq j$ and $\mathcal{A} \models Direct(\hat{\imath},\hat{\jmath}) \wedge \neg Big(\hat{\imath}) \wedge Big(\hat{\jmath})$, and

$r_4 =$ half of the number of (ordered) pairs $(i,j) \in D^2$ such that $i \neq j$ and $\mathcal{A} \models Direct(\hat{\imath},\hat{\jmath}) \wedge \neg Big(\hat{\imath}) \wedge \neg Big(\hat{\jmath})$,

then

$$\begin{aligned} w(\mathcal{A}) = &(3/8)^q (5/8)^{n-q} (2/3)^{r_1} (1/3)^{q(q-1)-r_1} (4/15)^{r_2} (11/15)^{q(n-q)-r_2} \\ &(4/15)^{r_3} (11/15)^{(n-q)q-r_3} (1/10)^{r_4} (9/10)^{(n-q)(n-q-1)-r_4} \end{aligned}$$

where the constants $3/8, 5/8, 2/3, 1/3$, etcetera, are motivated by the lifted Bayesian network in Figure 5.5.

Then we extend λ to a probability measure on the set of all subsets of $\mathbf{W}$ in the usual way:

$$\text{For every } \mathbf{X} \subseteq \mathbf{W}, \text{ let } \lambda(\mathbf{X}) = \sum_{\mathcal{A} \in \mathbf{X}} \lambda(\mathcal{A}).$$

According to the general definitions in Section 4.2.10 we let, for all L-sentences φ and ψ,

$$\mathbb{P}_\lambda(\varphi) = \lambda\big(\{\mathcal{A} \in \mathbf{W} : \mathcal{A} \models \varphi\}\big) \quad \text{and} \quad \mathbb{P}_\lambda(\varphi \mid \psi) = \frac{\mathbb{P}_\lambda(\varphi \wedge \psi)}{\mathbb{P}_\lambda(\psi)}.$$

By some basic (but slightly tedious) combinatorics one can show that for all distinct $i, j \in D$ we have:

$$\begin{aligned}
&\mathbb{P}_\lambda(Big(\hat{\imath})) = 3/8,\\
&\mathbb{P}_\lambda(Big(\hat{\imath}) \wedge Big(\hat{\jmath})) = \mathbb{P}_\lambda(Big(\hat{\imath})) \cdot \mathbb{P}_\lambda(Big(\hat{\jmath})),\\
&\mathbb{P}_\lambda(Direct(\hat{\imath}, \hat{\jmath}) |\ Big(\hat{\imath}) \wedge Big(\hat{\jmath})) = 2/3,\\
&\mathbb{P}_\lambda(Direct(\hat{\imath}, \hat{\jmath}) |\ Big(\hat{\imath}) \wedge \neg Big(\hat{\jmath})) = 4/15,\\
&\mathbb{P}_\lambda(Direct(\hat{\imath}, \hat{\jmath}) |\ \neg Big(\hat{\imath}) \wedge Big(\hat{\jmath})) = 4/15,\\
&\mathbb{P}_\lambda(Direct(\hat{\imath}, \hat{\jmath}) |\ \neg Big(\hat{\imath}) \wedge \neg Big(\hat{\jmath})) = 1/10.
\end{aligned}$$

Moreover, in the last four lines above, the probability is independent of whether $Direct(\hat{k}, \hat{l})$ holds or not if $k \neq i$ or $l \neq j$. Hence the (conditional) probabilities on atomic sentences correspond to the lifted Bayesian network in Figure 5.5.

Let $i, j \in D$ be any pair of distinct elements. We can view i and j as randomly chosen elements from D. Let us compute the probability that one can fly from i to j with at most one stop assuming that i and j are not big cities. The statement "i and j are not big" is expressed by the sentence

$$\neg Big(\hat{\imath}) \wedge \neg Big(\hat{\jmath})$$

and the statement "one can fly from i to j with at most one stop" is expressed by the sentence

$$\varphi := Direct(\hat{\imath}, \hat{\jmath}) \ \vee \ \exists z\big(Direct(\hat{\imath}, z) \wedge Direct(z, \hat{\jmath})\big).$$

Hence we want to compute $\mathbb{P}_\lambda\big(\varphi\ | \neg Big(\hat{\imath}) \wedge \neg Big(\hat{\jmath})\big)$. Since φ is logically equivalent to

$$\varphi' := Direct(\hat{\imath}, \hat{\jmath}) \ \vee \ \big(\neg Direct(\hat{\imath}, \hat{\jmath}) \wedge \exists z\big(Direct(\hat{\imath}, z) \wedge Direct(z, \hat{\jmath})\big)\big)$$

we have, by Proposition 4.2.5,

$$\mathbb{P}_\lambda\big(\varphi' \mid \neg Big(\hat{\imath}) \wedge \neg Big(\hat{\jmath})\big) = \mathbb{P}_\lambda\big(\varphi \mid \neg Big(\hat{\imath}) \wedge \neg Big(\hat{\jmath})\big).$$

Let $\varphi'' := \neg Direct(\hat{\imath}, \hat{\jmath}) \wedge \exists z\big(Direct(\hat{\imath}, z) \wedge Direct(z, \hat{\jmath})$. By Proposition 4.2.5 again, we have

$$\begin{aligned}
&\mathbb{P}_\lambda\big(\varphi' \mid \neg Big(\hat{\imath}) \wedge \neg Big(\hat{\jmath})\big) =\\
&\mathbb{P}_\lambda\big(Direct(\hat{\imath}, \hat{\jmath}) \mid \neg Big(\hat{\imath}) \wedge \neg Big(\hat{\jmath})\big) \ + \ \mathbb{P}_\lambda\big(\varphi'' \mid \neg Big(\hat{\imath}) \wedge \neg Big(\hat{\jmath})\big).
\end{aligned}$$

We immediately get $\mathbb{P}_\lambda\big(Direct(\hat{\imath}, \hat{\jmath}) \mid \neg Big(\hat{\imath}) \wedge \neg Big(\hat{\jmath})\big) = 1/10$.

Now consider arbitrary $k \in D \setminus \{i,j\}$. We have

$$\begin{aligned}
&\mathbb{P}_\lambda(Big(\hat{k})) = 3/8,\\
&\mathbb{P}_\lambda\big(Direct(\hat{\imath},\hat{k}) \mid Big(\hat{k}) \wedge \neg Big(\hat{\imath})\big) = 4/15,\\
&\mathbb{P}_\lambda\big(Direct(\hat{k},\hat{\jmath}) \mid Big(\hat{k}) \wedge \neg Big(\hat{\jmath})\big) = 4/15,\\
&\mathbb{P}_\lambda\big(Direct(\hat{\imath},\hat{k}) \mid \neg Big(\hat{k}) \wedge \neg Big(\hat{\imath})\big) = 1/10,\\
&\mathbb{P}_\lambda\big(Direct(\hat{k},\hat{\jmath}) \mid \neg Big(\hat{k}) \wedge \neg Big(\hat{\jmath})\big) = 1/10.
\end{aligned}$$

Hence (due to the independencies mentioned above),

$$\mathbb{P}_\lambda\big(Direct(\hat{\imath},\hat{k}) \wedge Direct(\hat{k},\hat{\jmath})\big) = \frac{3}{8}\cdot\frac{4}{15}\cdot\frac{4}{15} + \frac{5}{8}\cdot\frac{1}{10}\cdot\frac{1}{10} = \frac{79}{2400}.$$

It follows that the probability that $Direct(\hat{\imath},\hat{k}) \wedge Direct(\hat{k},\hat{\jmath})$ is *false* for *every* $k \in D \setminus \{i,j\}$ is $\left(1-\frac{79}{2400}\right)^{n-2}$. Thus the probability that $\exists z(Direct(\hat{\imath},z) \wedge Direct(z,\hat{\jmath}))$ is true is $1-\left(1-\frac{79}{2400}\right)^{n-2}$. It follows that

$$\mathbb{P}_\lambda\big(\varphi'' \mid \neg Big(\hat{\imath}) \wedge \neg Big(\hat{\jmath})\big) = \frac{9}{10}\Big(1-\Big(1-\frac{79}{2400}\Big)^{n-2}\Big)$$

and finally

$$\mathbb{P}_\lambda\big(\varphi' \mid \neg Big(\hat{\imath}) \wedge \neg Big(\hat{\jmath})\big) = \frac{1}{10} \ + \ \frac{9}{10}\Big(1-\Big(1-\frac{79}{2400}\Big)^{n-2}\Big).$$

Note that the above probability approaches 1 as n (the size of the domain) tends to infinity.

Now let us estimate the probability that between every pair of distinct small cities one can fly from the first city to the other with at most one stop. The statement "between every pair of distinct small cities one can fly from one to the other with at most one stop" is expressed by the sentence

$$\begin{aligned}
\psi := \forall x \forall y\big((x \neq y \wedge \neg Big(x) \wedge \neg Big(y)) \ \rightarrow\\
(Direct(x,y) \vee \exists z(Direct(x,z) \wedge Direct(z,y))\big).
\end{aligned}$$

The negation of this sentence, $\neg\psi$, is equivalent to

$$\exists x \exists y\big(x \neq y \wedge \neg Big(x) \wedge \neg Big(y) \wedge \neg Direct(x,y) \wedge \neg\exists z(Direct(x,z) \wedge Direct(z,y))\big).$$

From what we did above it follows that, conditioning on $\neg Big(\hat{\imath}) \wedge \neg Big(\hat{\jmath})$, the probability that

$$Direct(\hat{\imath},\hat{\jmath}) \vee \exists z(Direct(\hat{\imath},z) \wedge Direct(z,\hat{\jmath}))$$

is false is $\frac{9}{10}\left(1-\frac{79}{2400}\right)^{n-2}$. This holds for every choice of distinct $i,j \in D$. For each such choice, $\mathbb{P}_\lambda\big(\neg Big(\hat{\imath}) \wedge \neg Big(\hat{\jmath})\big) = (5/8)^2 = 25/64$. Since there are $n(n-1)$ choices of distinct $i,j \in D$ it follows that

$$\mathbb{P}_\lambda(\neg\psi) \ \leq \ n(n-1)\cdot\frac{25}{64}\cdot\frac{9}{10}\Big(1-\frac{79}{2400}\Big)^{n-2}.$$

Hence

$$\mathbb{P}_\lambda(\psi) \ > \ 1 \ - \ n(n-1)\cdot\frac{25}{64}\cdot\frac{9}{10}\Big(1-\frac{79}{2400}\Big)^{n-2}.$$

Again observe that $\mathbb{P}_\lambda(\psi)$ tends to 1 as n tends to infinity.

5.3 The case of partially observed data

Now we revisit the example in Section 5.2.1, but with the difference that the database has only partial information about what items (among I_1, I_2 and I_3) customers bought. Now the database is illustrated by the table in Figure 5.7 where we do not know if customers C and J bought item I_1 and we do not know if customer G bought item I_3. This is illustrated by blank slots in the corresponding places in the table.

Figure 5.7:

customer	buys I_1	buys I_2	buys I_3
A	yes	no	yes
B	no	yes	no
C		no	yes
D	yes	yes	yes
E	no	no	no
F	yes	no	yes
G	yes	yes	
H	no	no	no
I	no	yes	yes
J		yes	yes
K	no	no	yes
L	yes	yes	yes

Although we do not have any information about whether C or J bought item I_1 we may still want to find a reasonable estimate of the probability that a randomly chosen person from the database bought I_1. We can reason in the same way to estimate the probability that a randomly chosen customer from the database bought I_3. (We still have full information about who bought, or did not buy, I_2, so there is no need for estimating the probability that a randomly chosen customer from the database bought I_2; this probability can still be explicitly computed from the database as 1/2.)

I will describe an algorithm called the *Expectation Maximisation algorithm*, or *EM algorithm*, which converges to the so-called *maximum likelyhood estimate* (see for example [23, Definition 5.8.2]). The ideas underlying the reasoning below are useful in more general contexts. For more general (and abstract) accounts of the EM algorithm, see for example the following sources: [1, Section 8.5.1, p. 268–269], [3, Section 7.1.2.], [2], [17]. In the context considered here it turns out that the number (the maximum likelyhood estimate) that the EM algorithm converges to can be computed directly from the data given by the table, as stated by Proposition 5.3.1 below. The steps of the EM algorithm are nevertheless good to understand since they motivate why the estimate obtained is a reasonable estimate.

So let us estimate the probability that a random customer bought I_1, by using the EM algorithm.

Let X_A, $X_B, \ldots,$ X_L be binary random variables defined as follows for every $\Lambda \in \{A, B, \ldots, L\}$:

$$X_\Lambda = \begin{cases} 1 \text{ if } \Lambda \text{ bought } I_1, \\ 0 \text{ if } \Lambda \text{ did not buy } I_1. \end{cases}$$

We denote "the probability that X_Λ is 1" by "$w(X_\Lambda = 1)$". Since we know that A bought I_1 we let $w(X_A = 1) = 1$. Since we know that B did not buy I_1 we let $w(X_B = 1) = 0$.

We continue in the same way for all customers for which we know if they bought I_1 or not. It remains to consider the customers C and J for which we do not have information about whether they bought I_1 or not. We let $w(X_C = 1) = w(X_J = 1) = p_0$ for some initial "guess" $0 < p_0 < 1$. Now let

$$X = X_A + X_B + \ldots + X_L,$$

so X is the number of customers who bought I_1. Then, given the assigned probabilities, the expected number of customers who bought I_1 is

$$\begin{aligned}\mathbb{E}(X) &= \mathbb{E}(X_A) + \mathbb{E}(X_B) + \ldots + \mathbb{E}(X_L) \quad \text{(using linearity of expectation)}\\ &= w(X_A = 1) + w(X_B = 1) + \ldots + w(X_L = 1) = 5 + 2p_0.\end{aligned}$$

The number $\mathbb{E}(X) = 5 + 2p_0$ computed above is called the *expected count* (of customers who bought I_1). Since there are 12 customers in total, the next estimate of the probability that a random customer bought I_1 is now set to

$$p_1 = \frac{\mathbb{E}(X)}{12} = \frac{5 + 2p_0}{12}.$$

Now we can iterate the whole procedure again, using the improved guess p_1. This gives the new expected count $\mathbb{E}(X) = 5 + 2p_1$ and then the new estimate of the probability that a random customer bought I_1 is $p_2 = (5+2p_1)/12$. We can continue to iterate the procedure, each time getting a new estimate

$$p_{n+1} = \frac{5 + 2p_n}{12}$$

of the probability that a random customer bought I_1. No matter which $0 < p_0 < 1$ we choose, the sequence $p_0, p_1, p_2, \ldots$ converges to $1/2$. For example, if $p_0 = 0.1$, then $p_1 = 13/30$, $p_2 = 22/45$, $p_3 = 269/540 \approx 0.4981$.

The next proposition shows how we could have computed the maximum likelyhood estimate of the probability that a random customer bought I_1 directly.

Proposition 5.3.1 *Let (S, μ) be a probability space where S is finite and μ is the* counting measure*, i.e. $\mu(s) = 1/|S|$ for all $s \in S$. Let $X : S \to \{0, 1\}$ be a binary random variable and suppose that $S = S_1 \cup S_0 \cup S_u$, $X(s) = 1$ for all $s \in S_1$, $X(s) = 0$ for all $s \in S_0$ and suppose that we have no information about the value $X(s)$ if $s \in S_u$. If $S_u \neq S$ (so that we know the value $X(s)$ for at least one $s \in S$) then the EM algorithm converges and the estimate of $\mu(\{s \in S : X(s) = 1\})$ to which the EM algorithm converges is equal to $\frac{|S_1|}{|S_1 \cup S_0|}$.*

Proof. Let $s = |S|$, $s_1 = |S_1|$, and $s_u = |S_u|$. Suppose that $S_u \neq S$ and consequently $s_u/s < 1$. According to the EM algorithm, the initial estimate p_0 is chosen as some number in the interval $(0, 1)$. Once the estimate p_n is defined, the next estimate is set to be $p_{n+1} = \frac{s_1 + s_u p_n}{s} = \frac{s_1}{s} + \frac{s_u}{s} p_n$. It follows that

$$\begin{aligned}p_{n+1} &= \frac{s_1}{s} + \frac{s_u}{s} p_n = \frac{s_1}{s} + \frac{s_u}{s}\Big(\frac{s_1}{s} + \frac{s_u}{s} p_{n-1}\Big)\\ &= \frac{s_1}{s} + \frac{s_u}{s}\frac{s_1}{s} + \Big(\frac{s_u}{s}\Big)^2 p_{n-1} = \ldots\\ &= \frac{s_1}{s}\Big(1 + \frac{s_u}{s} + \Big(\frac{s_u}{s}\Big)^2 + \ldots + \Big(\frac{s_u}{s}\Big)^n + \Big(\frac{s_u}{s}\Big)^{n+1} p_0\Big).\end{aligned}$$

From the last expression we see (as $0 \leq \frac{s_u}{s} < 1$) that

$$\lim_{n \to \infty} p_{n+1} = \frac{s_1}{s} \cdot \frac{1}{1 - s_u/s} = \frac{s_1}{s - s_u} = \frac{s_1}{s_1 + s_0}.$$

□

In the probabilistic model that we construct we let the probability that a customer buys I_1 be 1/2 and we denote this by '$\mu(I_1(x)) = 1/2$', where, as in Section 5.2.1, $I_1(x)$ represents the statement "x buys I_1".

Now we would like to estimate the conditional probability that a random customer bought I_3, given that the customer bought I_1. Like in Section 5.2.1 we denote this probability by $\mu(I_3(x) \mid I_1(x))$. By definition of conditional probability, we should have

$$\mu(I_3(x) \mid I_1(x)) = \frac{\mu(I_3(x) \wedge I_1(x))}{\mu(I_1(x))}$$

where $\mu(I_3(x) \wedge I_1(x))$ denotes the probability that a randomly chosen customer bought items I_3 and I_1. We have already estimated $\mu(I_1(x))$ (to be 1/2) so it remains to estimate $\mu(I_3(x) \wedge I_1(x))$. By Proposition refthe number to which the EM algorithm converges it suffices to count the number of customers who bought the item I_1 *and* the item I_3 and divide it by the number of customers such that we know if the bought both these items or not. We see that A, D, F and L bought both I_1 and I_3. There are 9 customers such that we know if they bought both I_1 or I_3 or not. (For customers C, G and J we do not have enought information to determine if they bought both items.) Hence the EM estimate of $\mu(I_3(x) \wedge I_1(x))$ is 4/9 and we get

$$\mu(I_3(x) \mid I_1(x)) = \frac{\mu(I_3(x) \wedge I_1(x))}{\mu(I_1(x))} = \frac{4/9}{1/2} = 8/9.$$

The conditional probability $\mu(I_3(x) \mid I_1(x))$ that we estimated in this section is different from the conditional probability $\mu(I_3(x) \mid I_1(x))$ in Figure 5.2 of Section 5.2.1 (which was 1), but this should not be a surprise, since in this section we did not have full information about what the customers $A, B, \ldots, L$ bought so the estimatate of $\mu(I_3(x) \mid I_1(x))$ is an estimate based on what we do know about the customers and what – given this knowledge – is the most likely probability.

We can reason in the same way to compute the maximum likelyhood estimate 3/11 of the probability that a random customer buys both I_1 and I_2; with symbols, $\mu(I_2(x) \wedge I_1(x)) = 3/11$. Hence we get the conditional probability $\mu(I_2(x) \mid I_1(x)) = \mu(I_2(x) \wedge I_1(x)) \mid \mu(I_1(x)) = \frac{3/11}{1/2} = 6/11$. In a similar way we can find maximum likelyhood estimates of $\mu(I_2(x) \mid \neg I_1(x))$ and $\mu(I_3(x) \mid \neg I_1(x))$.

One also finds that the maximum likelyhood estimate of $\mu(I_3(x) \mid I_1(x) \wedge I_2(x))$ is $\frac{2/10}{3/11} = 22/30$ which is not close to $\mu(I_3(x) \mid I_1(x)) = 8/9$, so in the situation of this section, with only partial data as in the table in Figure 5.7, it is not reasonable to assume that $I_3(x)$ is independent from $I_2(x)$ over $I_1(x)$.

Exercise 5.3.2 Suppose that we have all the data in the table in Figure 5.1 in Section 5.2.1 *except* that we do not know if G bought item I_3 (so only this one piece of data is missing). Estimate $\mu(I_3(x) \mid I_1(x))$ and $\mu(I_3 \mid I_1(x) \wedge I_2(x))$. Is it reasonable to consider $I_3(x)$ to be independent from $I_2(x)$ over $I_1(x)$?

Chapter 6

Scalability: inference on large domains

6.1 Introduction

In sections 5.2.1 and 5.2.2 examples of inferences where made. More precisely, we considered a first-order sentence, call it φ in this discussion, and the probability that it is holds in a possible world/scenario with a given domain. In both examples we saw that the probability that φ holds could be described as an expression, call it $\alpha(n)$, with parameter n where n is the size of the domain (so what the domain actually is does not matter; only its size matters). In principle one could have derived this probability in the following way where we denote the domain by D and the set of all possible worlds (or formally, L-structures for a relevant language L) with domain D by $\mathbf{W}_D$: For each possible world $\mathcal{A} \in \mathbf{W}_D$, find out whether φ holds in $\mathcal{A}$ or not. Then compute

$$\mathbb{P}_D(\varphi) = \sum_{\substack{\mathcal{A}\in\mathbf{W},\\ \mathcal{A}\models\varphi}} \mathbb{P}(\mathcal{A}),$$

where $\mathbb{P}_D$ denotes the probability distribution on $\mathbf{W}_D$ determined by the given lifted Bayesian network as explained in the examples. However, for any language with at least one relation symbol, $|\mathbf{W}_D| \geq 2^n$ where $n = |D|$, so this method requires exponential time in the size of the domain, which is very inefficient if the domain is large.

But we did not proced like this in sections 5.2.1 and 5.2.2. Instead we analyzed the structure of the sentence φ and could, using the definition of $\mathbb{P}_D$, infer, or approximate, the value of $\mathbb{P}_D(\varphi)$ without considering every single $\mathcal{A} \in \mathbf{W}_D$ in detail. This method is much faster and note that it is even *independent of the domain size*, that is, the procedure goes through exactly the same steps no matter what n is. The procedure only depends on the structure (or complexity) of φ and of the lifted Bayesian network which is used to define the probability distribution on $\mathbf{W}_D$. This matters because in real world applications the size of the domain D of interest is normally much larger than the size of formulas φ and of the lifted graphical model which is used. For example the domain may include a large human population, or a large set of media messages, in which case its size may be in the hundreds of thousands, or millions. On the other hand, if the size of a logical formula is measured (for example) by its length as a string, then interesting events can be described by formulas of much less size, say in the hundreds or thousands. Similarly the size of a lifted Bayesian network (or other graphical model) will typically be much smaller than the size of the domain of interest.

In what follows we will see that the examples which we discussed above are just instances of a wider phenomenon which is of interest to the problem of *scalability*, that is, the problem of finding computationally efficent methods (algorithms) for making probabilistic inferences on varying and large domains. But first we need to make some definitions which are general enough to cover the above mentioned examples (and these definitions can be further generalized in various ways).

6.2 Some general results concerning scalability

When saying that a first-order language L is *finite and relational* we mean that it is finite and contains only relation symbols. Recall from Section 1.1.3 that $FO(L)$ denotes the set of all first-order formulas which can be formed by using the language L.

Definition 6.2.1 (Lifted Bayesian first-order network) Let L be a finite and relational first-order language. We define a *lifted Bayesian first-order network for L*, or an *$LBN(L)$-network*, to consist of the following components:

(a) A directed acyclic graph $\mathbb{G}$ with vertex set L. For every $R \in L$, let $\text{par}(R)$ denote the set of all parents of R in $\mathbb{G}$.

(b) For each $R \in L$, a number $\nu_R \in \mathbb{N}^+$, first-order L-formulas $\chi_{R,i}(\vec{x}) \in FO(\text{par}(R))$, for $i = 1, \ldots, \nu_R$, where $|\vec{x}|$ equals the arity of R, such that $\forall \vec{x}\big(\bigvee_{i=1}^{\nu_R} \chi_{R,i}(\vec{x})\big)$ is valid (i.e. true in all $\text{par}(R)$-structures) and if $i \neq j$ then $\exists \vec{x}\big(\chi_{R,i}(\vec{x}) \wedge \chi_{R,j}(\vec{x})\big)$ is unsatisfiable. Each $\chi_{R,i}$ will be called an *aggregation formula (of $\mathbb{G}$)*.

(c) For each $R \in L$ and each $1 \leq i \leq \nu_R$, a number denoted $\mu(R \mid \chi_{R,i})$ (or $\mu(R(\vec{x}) \mid \chi_{R,i}(\vec{x}))$) in the interval $[0, 1]$.

The intuitive meaning of $\mu(R \mid \chi_{R,i})$ in part (c) is that if $\vec{a}$ is a sequence of elements from a structure and $\vec{a}$ satisfies $\chi_{R,i}(\vec{x})$, then the probability that $\vec{a}$ satisfies $R(\vec{x})$ is $\mu(R \mid \chi_{R,i})$.

Remark 6.2.2 (i) We will use the same symbol, usually $\mathbb{G}$, for an $LBN(L)$-network and for its underlying DAG.
(ii) It will be convenient for technical reasons to sometimes consider an empty language L, that is a language which contains no relation symbol, function symbol or constant symbol. In this case, the underlying DAG of an $LBN(L)$-network has no vertex at all and hence no edge at all, and parts (b) and (c) of Definition 6.2.1 trivially hold.

Next we define, for an arbitrary finite domain D, how an $LBN(L)$-network induces a probability distribution on the set $\mathbf{W}_D$ of all L-structures with domain D.

Definition 6.2.3 (Subnetwork) Let $\mathbb{G}$ denote an $LBN(L)$-network where L is a finite and relational language. Suppose that $L' \subseteq L$ is such that if $R \in L'$ then $\text{par}(R) \subseteq L'$. Then it is easy to see that L' determines an $LBN(L')$-network $\mathbb{G}'$ for such that

- the vertex set of the underlying DAG of $\mathbb{G}'$ is L',
- for every $R \in L'$, the number ν_R and the formulas $\chi_{R,i}$, $i = 1, \ldots, \nu_R$, are the same as those for $\mathbb{G}$,
- for every $R \in L'$ and every $1 \leq i \leq \nu_R$, the numbers $\mu(R \mid \chi_{R,i})$ are the same as those for $\mathbb{G}$.

We call the so defined $LBN(L')$-network the *subnetwork (of $\mathbb{G}$) induced by L'.*

The notion defined below helps us to consider the different "levels" in a DAG. Vertices without parents are on the 0^{th} level.

Definition 6.2.4 Suppose that G is a DAG with nonempty and finite vertex set V.

(i) Let $a \in V$. We define the *maximal path rank of a*, or just *mp-rank of a*, denoted $\mathrm{mpr}(a)$, to be the length of the longest directed path having a as its last vertex (i.e. the largest k such that there is a sequence of vertices $a_0, a_1, \ldots, a_k$ where $a = a_k$ and (a_i, a_{i+1}) is a directed edge for each $i = 0, \ldots, k-1$).

(ii) The *maximal path rank of G*, or just *mp-rank of G*, denoted $\mathrm{mpr}(G)$, is defined as $\mathrm{mpr}(G) = \max\{\mathrm{mpr}(a) : a \in V\}$.

Observe that if G is a DAG with vertex set V, $\mathrm{mpr}(G) = r$ and G' is the induced subgraph of G with vertex set $V' = \{a \in V : \mathrm{mpr}(a) < r\}$, then, for every $a \in V'$, the mp-rank of a is the same no matter if we compute it with respect to G' or with respect to G; it follows that $\mathrm{mpr}(G') = r - 1$.

In the next definition we use some (standard) notation explained Section 1.1.3.

Definition 6.2.5 (The probability distribution induced by an $LBN(L)$-network) Let L be a finite and relational first-order language and let $\mathbf{W}_D$ be the set of all L-structures with domain D. Moreover, let $\mathbb{G}$ be an $LBN(L)$-network (whose underlying DAG is also denoted by $\mathbb{G}$).

(i) If L is empty, then $\mathbf{W}_D$ contains only one L-structure which is essentially the set D (without further structure). Then let $\mathbb{P}_D$ be the unique probability distribution on $\mathbf{W}_D$ which gives the unique structure in $\mathbf{W}_D$ probability 1.

(ii) Now suppose that L is non-empty. By induction on the mp-rank of the DAG of $\mathbb{G}$ we will define a probability distribution on $\mathbf{W}_D$. First we need some notation. Let ρ be the mp-rank of the underlying DAG of $\mathbb{G}$. For each $0 \leq r \leq \rho$ let $\mathbb{G}_r$ be the subnetwork induced by $L_r = \{R \in L : \mathrm{mpr}(R) \leq r\}$ and note that $\mathbb{G}_\rho = \mathbb{G}$. Also let L_{-1} denote the empty language and let $\mathbb{P}_D^{-1}$ be the unique probability distribution on the set of L_{-1}-structures with domain D which we denote $\mathbf{W}_D^{-1}$. By induction on r we define, for every $r = 0, 1, \ldots, \rho$, a probability distribution $\mathbb{P}_D^r$ on the set $\mathbf{W}_D^r$ of all L_r-structures with domain D as follows: For every $\mathcal{A} \in \mathbf{W}_D^r$,

$$\mathbb{P}_D^r(\mathcal{A}) \;=\; \mathbb{P}_D^{r-1}(\mathcal{A}{\restriction}L_{r-1}) \prod_{R \in L_r \setminus L_{r-1}} \prod_{i=1}^{\nu_R} \prod_{\vec{a} \in \chi_{R,i}(\mathcal{A}\restriction L_{r-1})} \lambda(\mathcal{A}, R, i, \vec{a})$$

where

$$\lambda(\mathcal{A}, R, i, \vec{a}) = \begin{cases} \mu(R \mid \chi_{R,i}) & \text{if } \mathcal{A} \models \chi_{R,i}(\vec{a}) \wedge R(\vec{a}), \\ 1 - \mu(R \mid \chi_{R,i}) & \text{if } \mathcal{A} \models \chi_{R,i}(\vec{a}) \wedge \neg R(\vec{a}), \\ 0 & \text{otherwise.} \end{cases}$$

Finally we let $\mathbf{W}_D = \mathbf{W}_D^\rho$ and $\mathbb{P}_D = \mathbb{P}_D^\rho$, so $\mathbb{P}_D$ is a probability distribution on $\mathbf{W}_D$

The above definition generalizes the probability distributions considered in the examples. Note that the nature of the objects in the domain D is irrelevant. Also, the definition of $\mathbb{P}_D$ works out no matter what the size of D is. (But of course, if the $LBN(L)$-network is supposed to model, for example, the various properties and relationships on the domain of media and markets, then it would not make sense to apply it to a domain of, for example, DNA-sequences and diseases.)

Notation 6.2.6 (Renaming $\mathbf{W}_D$ and $\mathbb{P}_D$) Since the nature of the elements/objects in the domain D has no relevance in Definition 6.2.5, we will from now on assume that $D = [n]$ where $[n] = \{1, \ldots, n\}$. We also rename $\mathbf{W}_D$ and $\mathbb{P}_D$ by $\mathbf{W}_n$ and $\mathbb{P}_n$, respectively. With this notational convention we now have a sequence of pairs $(\mathbf{W}_n, \mathbb{P}_n)$, $n \in \mathbb{N}^+$, where $\mathbb{P}_n$ is a probability distribution on $\mathbf{W}_n$.

Definition 6.2.7 For every $\varphi(\vec{x}) \in FO(L)$, every $n \in \mathbb{N}^+$ and every $\bar{a}[n]^{|\vec{x}|}$, we let

$$\mathbb{P}_n(\varphi(\vec{a})) := \mathbb{P}_n\big(\{\mathcal{A} \in \mathbf{W}_n : \mathcal{A} \models \varphi(\vec{a})\}\big),$$

so $\mathbb{P}_n(\varphi(\vec{a}))$ denotes the probability that $\varphi(\vec{a})$ is satisfied by a random $\mathcal{A} \in \mathbf{W}_n$.

The following basic result can be proved in a similar manner as Lemma 4.6 in [21].

Lemma 6.2.8 *Let L be a finite and relational language and $\mathbb{G}$ an $LBN(L)$-network. Also let $\mathbb{P}_n$ be the probability distribution induced by $\mathbb{G}$ on $\mathbf{W}_n$ according to Definition 6.2.5. If $\varphi(\vec{x}) \in FO(L)$ and $\vec{a}, \vec{b} \in [n]^{|\vec{x}|}$ then $\mathbb{P}_n(\varphi(\vec{a})) = \mathbb{P}_n(\varphi(\vec{b}))$.*

Now we can state a convergence law for the sequence of probability distributions $(\mathbf{W}_n, \mathbb{P}_n)$ which are induced by an $LBN(L)$-network $\mathbb{G}$. Then we discuss its relevance for scalability.

Theorem 6.2.9 (Convergence law) *Let L be a finite and relational signature, let $\mathbb{G}$ be an $LBN(L)$-network and, for each $n \in \mathbb{N}^+$, let $\mathbb{P}_n$ be the probability distribution induced by $\mathbb{G}$ (according to Definition 6.2.5) on the set $\mathbf{W}_n$ of all L-structures with domain $[n]$. For every $\varphi(\vec{x}) \in FO(L)$ there are $c > 0$ and $0 \leq d \leq 1$, depending only on $\varphi(\vec{x})$ and $\mathbb{G}$, such that for every $n \in \mathbb{N}^+$ and every $\vec{a} \in [n]^{|\vec{x}|}$,*

$$\big|\mathbb{P}_n(\varphi(\vec{a})) - d\big| \ \leq \ e^{-cn}.$$

Moreover, if φ has no free variable (i.e. is a sentence), then $\mathbb{P}_n(\varphi)$ converges to either 0 or 1.

Theorem 6.2.9 follows from a result by Jaeger from 1998 [13, Theorem 3.9] although this may not be immediately apparent because Jaeger uses a different type of Bayesian network which he calls *Relational Bayesian Network* which uses so-called “probability formulas” which can take arbitrary values in the unit interval $[0, 1]$. On the other hand, Theorem 6.2.9 is an apparent special case of [20, Theorem 3.15][1] because this result considers a more general logic $CPL(L)$ where $FO(L) \subseteq CPL(L)$ and every formula in $FO(L)$ is “non-critical” in the sense of [20].

[1]There is an unfortunate typo in [20, Theorem 3.15] where the inequality ‘$|\mathbb{P}_n(\varphi(\vec{a})) - d| \leq 1 - e^{-cn}$’ should read ‘$|\mathbb{P}_n(\varphi(\vec{a})) - d| \leq e^{-cn}$’.

Implications of Theorem 6.2.9 on random sampling to estimate probabilities

The relevance of Theorem 6.2.9 for scalability is this. Let $\varphi(\vec{x}) \in FO(L)$, $m \in \mathbb{N}^+$ and $\vec{a} \in [m]^{|\vec{x}|}$. We want to estimate $\mathbb{P}_n(\varphi(\vec{a}))$ for some large n. By Theorem 6.2.9, $\lim_{n\to\infty} \mathbb{P}_n(\varphi(\vec{a}))$ exists, so let α denote this limit. Suppose that an error marginal $\varepsilon > 0$ is given. By the law of large numbers there is an integer k (depending on ε) such that, for any large enough n, if we take k random samples $\mathcal{A}_1, \ldots, \mathcal{A}_k \in \mathbf{W}_n$, then, with probability at least $1-\varepsilon$, the number of samples $\mathcal{A}_i$ such that $\mathcal{A}_i \models \varphi(\vec{a})$, divided by k, is in the interval $[\alpha-\varepsilon, \alpha+\varepsilon]$. To compute the truth value of $\varphi(\vec{a})$ in $\mathcal{A}_i$ we need to consider a domain of size n. But at least this can be done in time bounded by $\mathcal{O}(n^p)$, where p is the *quantifier-rank* of $\varphi(\vec{x})$. Moreover, the estimate thus obtained is, with probability at least $1-\varepsilon$, valid for every $n' \geq n$. (More precisely, if we consider any $n' > n$ then, with probabability at least $1-\varepsilon$, we get an estimate in the interval $[\alpha-\varepsilon, \alpha+\varepsilon]$.)

However, Theorem 6.2.9 does not give the whole truth and in fact it is possible determine $\lim_{n\to\infty} \mathbb{P}_n(\varphi(\vec{a}))$ by a procedure which is completely independent of n. The procedure depends on the following result which is a special case of [20, Theorem 3.14]:

Theorem 6.2.10 (Almost sure elimination of quantifiers) *Let L be a finite and relational signature, let $\mathbb{G}$ be a lifted Bayesian network and, for each $n \in \mathbb{N}^+$, let $\mathbb{P}_n$ be the probability distribution induced by $\mathbb{G}$ on $\mathbf{W}_n$ (according to Definition 6.2.5). For every $\varphi(\vec{x}) \in FO(L)$ there are a quantifier-free formula $\varphi^*(\vec{x}) \in FO(L)$ and $c > 0$, which depend only on $\varphi(\vec{x})$ and $\mathbb{G}$, such that for all sufficiently large n*

$$\mathbb{P}_n\big(\forall \vec{x}(\varphi(\vec{x}) \leftrightarrow \varphi^*(\vec{x}))\big) \;\geq\; 1 - e^{-cn}.$$

The proof of Theorem 6.2.10 is constructive in the sense that it shows how to find a quantifier-free formula $\varphi^*(\vec{x})$ such that

$$\lim_{n\to\infty} \mathbb{P}_n\big(\forall \vec{x}(\varphi(\vec{x}) \leftrightarrow \varphi^*(\vec{x}))\big) = 1.$$

Moreover, $\varphi^*(\vec{x})$ can be constructed so that every relation symbol in it also occurs in $\varphi(\vec{x})$. Since Theorem 6.2.10 is applicable to every aggregation formula $\chi_{R,i}(\vec{x})$ of $\mathbb{G}$ (as in Definition 6.2.1) we can can replace every aggregation formula $\chi_{R,i}(\vec{x})$ by a quantifier-free aggregation formula $\chi^*_{R,i}(\vec{x})$ which is "almost surely equivalent" to $\chi_{R,i}(\vec{x})$ and thus get a "simpler" $LBN(L)$-network which gives the same predictions asymptotically (i.e. up to an arbitrarily small error). More precisely, we have the following result which is a special case of [20, Theorem 3.16]:

Theorem 6.2.11 (An asymptotically equivalent "quantifier-free" network) *Let L, $\mathbb{G}$ and $\mathbb{P}_n$ be as in Theorem 6.2.10. Then for every aggregation formula $\chi_{R,i}(\vec{x})$ of $\mathbb{G}$ (as in Definition 6.2.1) there is a quantifier-free formula $\chi^*_{R,i}(\vec{x})$ containing only relation symbols that occur in $\chi_{R,i}$ such that if $\mathbb{G}^*$ is the lifted Bayesian network*

- *with the same underlying DAG as $\mathbb{G}$,*
- *where, for every $R \in \sigma$ and every $1 \leq i \leq \nu_R$, the aggregation formula $\chi_{R,i}$ is replaced by $\chi^*_{R,i}$, and*
- *where $\mu^*(R \mid \chi^*_{R,i}) = \mu(R \mid \chi_{R,i})$ for every $R \in \sigma$ and every $1 \leq i \leq \nu_R$,*

then for every $\varphi(\vec{y}) \in FO(\sigma)$ there is $d > 0$, depending only on $\varphi(\vec{y})$ and $\mathbb{G}$, such that for every $n \in \mathbb{N}^+$ and every $\vec{a} \in [n]^{|\vec{y}|}$, ,

$$\left|\mathbb{P}_n(\varphi(\vec{a})) - \mathbb{P}^*_n(\varphi(\vec{a}))\right| \;\leq\; e^{-dn},$$

where $\mathbb{P}^*_n$ is the the probability distribution on $\mathbf{W}_n$ according to Definition 6.2.5 if $\mathbb{G}$ is replaced by $\mathbb{G}^*$ and $\mathbb{P}_n$ is replaced by $\mathbb{P}^*_n$.

Remark 6.2.12 The proof of Theorem 6.2.11 (in [20]) actually shows that there is $d > 0$ such that for all sufficiently large n and all $\mathbf{X} \subseteq \mathbf{W}_n$, $|\mathbb{P}_n(\mathbf{X}) - \mathbb{P}^*_n(\mathbf{X})| \leq e^{-dn}$.

An $LBN(L)$-network as $\mathbb{G}^*$ in the above theorem, with only quantifier-free aggregation formulas, is computationally easier to handle than an $LBN(L)$-network in general, as the following basic result states.

Lemma 6.2.13 *Let $\mathbb{G}^*$ be an $LBN(L)$-network with only quantifier-free aggregation formulas and let $\mathbb{P}^*_n$ be the probability distribution on $\mathbf{W}_n$ which is induced by $\mathbb{G}^*$. Then, for every quantifier-free $\varphi(\vec{x}) \in FO(L)$, every $n \in \mathbb{N}^+$, and every $\vec{a} \in [n]^{|\vec{x}|}$, $\mathbb{P}^*_n(\varphi(\vec{a}))$ can be computed by using only $\varphi(\vec{x})$ and $\mathbb{G}^*$, so in particular the computation and the value of $\mathbb{P}^*_n(\varphi(\vec{a}))$ does not depend on n or $\vec{a}$.*

We will not prove Lemma 6.2.13, but only make a small remark on its proof. Every quantifier-free formula $\varphi(\vec{x})$ is equivalent to a formula on the form $\bigvee_{i=1}^{s} \theta_i(\vec{x})$ where each θ_i is a conjunction of literals and if $i \neq j$ then $\exists \vec{x}(\theta_i(\vec{x}) \wedge \theta_j(\vec{x}))$ is unsatisfiable. Then $\mathbb{P}^*_n(\varphi(\vec{a})) = \sum_{i=1}^{s} \mathbb{P}^*_n(\theta_i(\vec{a}))$ and, for each i, $\mathbb{P}^*_n(\theta_i(\vec{a}))$ can, by Definition 6.2.5 and since every aggregation formula of $\mathbb{G}^*$ is assumed to be quantifier-free, be computed by using only $\theta_i(\vec{x})$ and $\mathbb{G}^*$.

A procedure for probabilistic inference which is independent of the domain size

Let $\mathbb{G}$ be an $LBN(L)$-network, possibly with aggregation formulas which are not quantifier-free. Let $\varphi(\vec{x}) \in FO(L)$ and suppose that we want to estimate $\mathbb{P}_n(\varphi(\vec{a}))$ for some large n and $\vec{a} \in [n]$. By Theorem 6.2.10, there is a quantifier-free $\varphi^*(\vec{x}) \in FO(L)$ such that for all sufficiently large n

$$\mathbb{P}_n\big(\forall \vec{x}(\varphi(\vec{x}) \leftrightarrow \varphi^*(\vec{x}))\big) \;\geq\; 1 - e^{-cn},$$

for some constant $c > 0$. Hence

$$\mathbb{P}_n\big((\varphi(\vec{a}) \wedge \neg\varphi^*(\vec{a})) \vee (\neg\varphi(\vec{a}) \wedge \varphi^*(\vec{a}))\big) \;<\; e^{-cn}$$

and consequently

$$\left|\mathbb{P}_n(\varphi(\vec{a})) - \mathbb{P}_n(\varphi^*(\vec{a}))\right| \;<\; e^{-cn}. \tag{6.2.1}$$

Moreover, as mentioned after Theorem 6.2.10, the formula $\varphi^*(\vec{x})$ can be constructed by using only $\varphi(\vec{x})$ and $\mathbb{G}$.

By Theorem 6.2.11 there is an $LBN(L)$-network $\mathbb{G}^*$ with the properties described in that theorem, so in particular

$$\left|\mathbb{P}_n(\varphi^*(\vec{a})) - \mathbb{P}^*_n(\varphi^*(\vec{a}))\right| \;<\; e^{-dn}, \tag{6.2.2}$$

for some constant $d > 0$. Also, the $LBN(L)$-network $\mathbb{G}^*$ can be computed from (only) $\mathbb{G}$, since it is obtained by replacing every aggregation formula of $\mathbb{G}$ by an "almost surely equivalent" quantifier-free formula by using Theorem 6.2.11. From (6.2.1) and (6.2.2) we get

$$\big|\mathbb{P}_n(\varphi(\vec{a})) - \mathbb{P}^*_n(\varphi^*(\vec{a}))\big| \ < \ e^{-cn} + e^{-dn}.$$

By Lemma 6.2.13, $\alpha = \mathbb{P}^*_n(\varphi^*(\vec{a}))$ does not depend on n or $\vec{a}$, but can be computed (only) from $\varphi^*(\vec{x})$ and $\mathbb{G}^*$. So if an error marginal $\varepsilon > 0$ is given, then, for all sufficiently large n, $|\mathbb{P}_n(\varphi(\vec{a})) - \alpha| < \varepsilon$. Thus we get can compute an arbitrarily good estimate (for sufficiently large n) of $\mathbb{P}_n(\varphi(\vec{a}))$ by using $\varphi(\vec{x})$ and $\mathbb{G}$ as input to the algorithm, where the later two objects are independent of the size n of the domain on which we want to make inferences.

6.3 Generalizations and other approaches

6.3.1 Other logics

The previous section considered first-order logic as the formal language for describing events. While first-order logic can describe many type of events of interest it also has its limitatons. We may, for example, want to describe the event that "at least 30% of the population is infected by disease D", or the event that "for at least 60% of all individuals x, if at least 30% of the contacts of person x are infected by disease D, then x is infected by D". These events cannot be described with $FO(L)$ for any language L, but they can be described by formulas in $CPL(L)$, where CPL stands for "conditional probability logic", which is studied in [20]. As another example, with CPL one can also express that two random variables (expressed by CPL-formulas) are are (conditionally) independent from each other. Definition 6.2.1 can be generalized by allowing aggregation formulas to be $CPL(L)$-formulas. Then Theorem 6.2.9, 6.2.10 – 6.2.11 still hold if all references to formulas concern *non-critical* $CPL(L)$*-formulas* and this is, from a practical perspective, a quite mild restriction. Also the implications for scalability hold for $CPL(L)$ with the same arguments as in the previous section for $FO(L)$.

In analysis of data one often uses so-called *aggregation functions* (also called aggregate functions or combination functions). An aggregation function takes a finite sequence of real numbers and returns a real number. Important examples of aggregation functions include the *maximum* (of the numbers in the sequence), the *minimum*, the *arithmetic mean*, the *geometric mean*, and *noisy-or*, and one can use these to construct more complex aggregation functions which are also of interest. For example, let $\varphi(x) \in FO(L)$ for some language L and let $\mathcal{A}$ be a finite L-structure. We may be interested in the proportion of $a \in A$ such that $\mathcal{A} \models \varphi(a)$. Suppose $A = \{a_1, \ldots, a_n\}$. Then this proportion is obtained by taking the arithmetic mean of the sequence $(k_1, \ldots, k_n)$ where $k_i = 1$ if $\mathcal{A} \models \varphi(a_i)$ and $k_i = 0$ otherwise. Formally we could say, for example, that $\mathrm{am}(\varphi(x) : x)$ is a new formula, where 'am' denotes the arithmetic mean, and the interpretation of this new formula in a finite structure $\mathcal{A}$ is the arithmetic mean of the sequence of numbers $(k_1, \ldots, k_n)$ defined as above. But then the interpretation of $\mathrm{am}(\varphi(x) : x)$ in $\mathcal{A}$ need not be 0 or 1, but could be 1/3, for example. Thus the incorporation of aggregation functions into the formal logical language leads to many-valued logic.

Examples of many-valued logics of relevance in statistical relational learning include Jaegers probability logic in [13] and the probability logic denoted PLA (for "probability logic with aggregation functions") in work of myself and Weitkämper [21, 22]. Both of these logics use aggregation functions instead of quantifiers. The aggregation functions 'maximum' and 'minimum' can take the roles of '$\exists$' and '$\forall$' for formulas that only take the values 0 or 1. In both of these logics the ("truth") values of formulas are always in the unit interval $[0, 1]$. One can also define a "lifted", or "parametrized", Bayesian network appropriate for a logic with truth values in the unit interval in a rather natural way, as done in [13, 21, 22]. Moreover, in [13, 21, 22] different kinds of probabilistic convergence results are proved, which have relevance for scalability.

6.3.2 Other graphical models

The graphical models discussed in this text so far have been some form of Bayesian network, so conditional dependencies have been represented by a directed acyclic graph. Although a Bayesian network can always be learned from a dataset as explained in Section 5.1, one may get different Bayesian networks, depending on the order in which the random variables (formulas) of the network are listed in the construction algorithm. It does not necessarily happen that one finds an order where conditional dependencies and independencies are sufficently pronounced that a directed graphical model is the most natural choice of model. Instead one may choose to use a probabilistic model where conditional (in)dependencies are represented by undirected edges, as in the various existing variants of Markov networks. Then probabilistic dependencies are modelled, but no “direction” of the dependencies is given.

A (lifted) Markov network $\mathbb{G}$ can be used to determine a probability distribution $\mathbb{P}_D$ on the set of possible worlds (i.e. L-structures for some language L) with domain D, where D may be an arbitrary finite set. But unlike the various kinds of lifted Bayesian networks considered in this text, if D and D' are domains of different sizes it is in general unclear how $\mathbb{P}_D(\varphi(\vec{a}))$ and $\mathbb{P}_{D'}(\varphi(\vec{a}'))$ are related even if $\varphi(\vec{x})$ is an atomic formula and $\vec{a} \in D^{|\vec{x}|}$ and $\vec{a}' \in D'^{|\vec{x}|}$. To overcome this problem the notions of *domain-aware Markov logic network* and *domain-aware relational logistic regression network* (the later of which is a directed model, but of different nature than Bayesian networks) have been deviced and some results concerning convergence in probability (as the domain size tends to infinity) of events defined by quantifier-free formulas have been proved [29, 25, 32]. Related results concerning inference across different domains are found in [15, 14, 33].

Bibliography

[1] Luc De Raedt, *Logical and Relational Learning*, Springer-Verlag (2008).

[2] Luc De Raedt, Kristian Kersting, Probabilistic logic learning, *ACM SIGKDD Explorations Newsletter*, Vol. 5 (2003) 31–48.

[3] Luc De Raedt, Kristian Kersting, Sriraam Natarajan, David Poole, *Statistical Relational Artificial Intelligence: Logic, Probability, and Computation*, Synthesis Lectures on Artificial Intelligence and Machine Learning #32, Morgan & Claypool Publishers (2016).

[4] Pedro Domingos, Daniel Lowd, Unifying logical and statistical AI with Markov logic, *Communications of the ACM*, Vol. 62 (2019) 74–83.

[5] Heinz-Dieter Ebbinghaus, Jörg Flum, *Finite Model Theory*, Second Edition, Springer Verlag (1999).

[6] Haim Gaifman, Concerning measures in first order calculi, *Israel Journal of Mathematics*, Vol. 2 (1964) 1–18.

[7] Theodore Hailperin, Probability logic, *Notre Dame Journal of Formal Logic*, Vol. 25 (1984) 198–212.

[8] Joseph Y. Halpern, An analysis of first-order logics of probability, *Artificial Intelligence*, Vol. 46 (1990) 311–350.

[9] Lise Getoor, Ben Taskar (Editors), *Introduction to Statistical Relational Learning*, The MIT Press (2007).

[10] Shawn Hedman, *A First Course in Logic: An Introduction to Model Theory, Proof Theory, Computability, and Complexity*, Oxford University Press (2004).

[11] U. Hustadt, R. A. Schmidt, L. Georgieva, A survey of decidable first-order fragments and description logics, *Journal on Relational Methods in Computer Science*, Vol. 1 (2004) 251 – 276.

[12] Tapani Hyttinen, Gianluca Paolini, Jouko Väänänen, A logic for arguing about probabilities in measure teams, *Archive for Mathematical Logic*, Vol. 56 (2017) 475–489.

[13] Manfred Jaeger, Convergence results for relational Bayesian networks, *Proceedings of the 13th Annual IEEE Symposium on Logic in Computer Science (LICS 98)* (1998).

[14] M. Jaeger, O. Schulte, A complete characterization of projectivity for statistical relational methods, in C. Bessiere (Ed.) *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, Yokohama, Japan, January 2020*, IJCAI, 4283–4290.

[15] D. Jain, A. Barthels, M. Beetz, Adaptive Markov logic networks: learning statistical relational models with dynamic parameters, in H. Coelho, R. Studer, M. J. Woolridge (Eds.), *ECAI 2010 – 19th European Conference on Artificial Intelligence, Lisbon, Portugal, August 16–20, 2010,* Frontiers in Artifical Intelligence and Applications, Vol. 215 (2010) 937–942, IOS Press, Amsterdam.

[16] Kristian Kersting, *An Inductive Logic Programming Approach to Statistical Relational Learning*, IOS Press (2006).

[17] Angelika Kimming, Lilyana Mihalkova, Lise Getoor, Lifted graphical models: a survey, *Machine Learning*, Vol. 99 (2015) 1–45.

[18] Daphne Koller, Nir Friedman, *Probabilistic graphical models: principles and techniques*, Massachusetts Institute of Technology (2009).

[19] Stephen C. Kleene, *Introduction to Metamathematics*, Wolters-Noordhoff Publishing – Groningen – North-Holland Publishing Company – Amsterdam (1971).

[20] Vera Koponen, Conditional probability logic, lifted Bayesian networks, and almost sure quantifier elimination, *Theoretical Computer Science*, Vol. 848 (2020) 1–27.

[21] Vera Koponen, Felix Weitkämper, Asymptotic elimination of partially continuous aggregation functions in directed graphical models, *Information and Computation*, Vol. 293 (2023) 105061.

[22] Vera Koponen, Felix Weitkämper, On the relative asymptotic expressivity of inference frameworks, submitted, preprint online: `https://arxiv.org/abs/2204.09457`

[23] Richard J. Larsen, Morris L. Marx, *An Introduction to Mathematical Statistics and its Applications*, Second Edition, Prentice-Hall (1986).

[24] Harry R. Lewis, Christos H. Papadimitriou, *Elements of the Theory of Computation*, Second Edition, Prentice-Hall (1998).

[25] H. Mittal, A. Bhardwaj, V. Gogate, P. Singla, Domain-size aware Markov logic networks, in K. Chaudhuri, M. Sugiyama (eds.) *The 22nd International Conference on Artificial Intelligence and Statistics, AISTATS 2019, Naha, Japan, 16—18 April 2019*, Proceedings of machine learning research, Vol. 89 (2019) 3216–3224.

[26] Jean-Francois Monin, *Understanding Formal Methods*, Springer-Verlag (2003).

[27] Judea Pearl, *Causality: Models, Reasoning, and Inference*, Second Edition, Cambridge University Press (2009).

[28] Christos H. Papadimitriou, *Computational Complexity*, Addison Wesley Longman (1995).

[29] D. Poole, D. Buchanan, S. M. Kazemi, K. Kersting, S. Natarajan, Population size extrapolation in relational probabilistic modelling, in U. Straccia, A. Cali (Eds.), *Scalable Uncertainty Management – 8th International Conference*, Oxford, UK, September 15–17 2014, Lecture Notes in Computer Science, Vol. 8720 (2014) 292–305.

[30] Stuart Russell, Unifying Logic and Probability, *Communications of the ACM*, Vol. 58 (2015) 88–97.

[31] Michael Sipser, *Introduction to the Theory of Computation*, Cengage Learning (2013).

[32] F. Weitkämper, Scaling the weight parameters in Markov logic networks and relational logistic regression models, `https://arxiv.org/abs/2103.15140`

[33] F. Weitkämper, An asymptotic analysis of probabilistic logic programming with implications for expressing projective families of distributions, `https://arxiv.org/abs/2102.08777`