TUHH
Technische Universität Hamburg-Harburg

Probabilistic Graphical Models
- Studienarbeit -

Florian Meyer - 20523857
Florian.Meyer@tu-harburg.de

Betreuer: Prof. Dr. rer. nat. habil. Ralf Möller
Institute for Software Systems
Hamburg University Of Technology

I, Florian Meyer, born 25.10.1985, solemnly declare that I have written this master thesis independently, and that I have not made use of any aid other than those acknowledged in this master thesis. Neither this master thesis, nor any other similar work, has been previously submitted to any examination board.

Florian Meyer

Contents

Acronyms

BN Bayesian Network.

BP Belief Propagation.

FOL First Order Logic.

i.i.d. independent and identically distributed.

KB Knowledge Base.

KBMC Knowledge Based Model Construction.

LBP Lifted Belief Propagation.

MAP maximum a posteriori.

ML Markov Logic.

MLN Markov Logic Network.

MN Markov Network.

RMN Relational Markov Network.

SRL Statistical Relational Learning.

List of Symbols

λ	Sample space
n	Natural number
e	Event
P	Probability Function
$\mathbf{F}$, $\mathbf{E}$, $\mathbf{G}$	Sets of Events
Y, X, D	Random variables
$\mathbf{Y}$, $\mathbf{X}$, $\mathbf{D}$	Set of Random variables
$\mathbf{P}$	Probability distribution
Ω_X	Sample space of random variable X
X_V	Vector with random variables as elements
E	Set of edges
G	Undirected graph
u, v	Single vertices in an undirected graph
$\mathcal{C}$	Set of cliques
C	Clique
ϕ_C	Clique potential for clique C
$\mathcal{X}$	Set of all possible assignments of values to a Markov Network's variables
Z	Partition function
$f_i(X)$	Feature function
$\mathcal{H}$	Markov Network
w	A real number
β	Normalizing constant
A, B	Logical sentences
M	A model in Logic
o	An object in Logic
$f(\lambda)$	Template feature
M_{RMN}	Relational Markov Network
Λ	Set of template features
κ	Object skeleton
$\alpha(\lambda)$	Signature of λ
$\mathcal{P}_{\kappa}^{M_{RMN}}$	Ground Gibbs distribution
γ	Assignment of values to logical variables
Γ_{κ}	Set of assignments to logical variables

$g(x)$	Superfeature
$\mathcal{X}_{KB}$	Set of worlds that satisfy KB
ED	Evidence database
$\mathbf{L}$	Markov Logic Network
F_i	Formula in First Order Logic
$\mathbf{C}$	Set of constants
c	Constant
x	Possible world
$\mathcal{C}_T$	Set of clique templates
f_c	Grounded feature

Chapter 1

Introduction

1.1 Motivation

Real world applications that require to reason over uncertain domains, such as in medical diagnosis, natural language processing systems or applications in the field of computational biology are successfully modeled with probabilistic graphical models like Bayesian Networks or Markov Networks. However all these applications have in common that the data that is used in these areas is of a relational nature, meaning that the objects in the domain interact with other objects or are in some other relation to each other. When applying models that assume non independent and identically distributed (i.i.d.) data, the data is flattened, accordingly the relational structure is lost. Moreover this is equivalent to a loss of information that could be used to generate more accurate results when working with this data. As a consequence it is an active field of research to find a way to model the relational structure of the data with probabilistic graphical models.

First Order Logic (FOL) is a very powerful language to represent and to reason over relational data. However modeling systems in FOL is very hard, and in some cases, when the systems get too complex, even infeasible to be performed by the user due to tight modeling constraints. Consequently, the idea is to combine the benefits of FOL and probabilistic graphical models in one formalism that exploits their advantages. This field of research is called Statistical Relational Learning [13, p.3].

In the last two decades a wide variety of systems has been proposed to unify the two formalisms. In this work two more recent ones, Markov Logic and Relational Markov Networks are presented. These two share many characteristics that are common among many other approaches to the Statistical Relational Learning (SRL) problem. The aim of this work is to provide a concise introduction to the field of SRL to the reader. This introduction covers representation as well as inference in the two formalisms, it discusses limitations and benefits of each model, and shows how the models are related.

1.2 Structure

First an introduction of necessary foundations essential to understand approaches to SRL is given in Chapter 2. We begin with basics of probability calculus, logic and probabilistic graphical models. Having explained all the prerequisites, Relational Markov Networks (RMNs) and Markov Logic (ML) are introduced and discussed in Chapter 3. Moreover improvements to inference and possible extensions are subject of Chapter 4. The work ends with a description of the connection between the models as well as showing the reader an outline of possible future work in this field.

Chapter 2

Foundations

2.1 Probability Calculus

In the next section an introduction to probability calculus is provided. This introduction is inspired by [30][Chapter 1.1] and provides the most fundamental concepts of modern probability theory, necessary to understand the concepts elaborated in the rest of this work.

2.1.1 Introduction

Probability theory deals with experiments that have a defined set of outcomes. The collection of all outcomes of an experiment is called the sample space. A sample space can be finite or infinite. Subsequently only the finite case will be considered. Every subset of the sample space is called an event, and if the cardinality of the event set is one, it is called elementary event.

Definition 2.1.1 ([30][p.4])**.** Suppose Ω is a sample space containing n elements. That is,

$$\Omega = \{e_1, e_2, \dots, e_n\}.$$

A function that assigns a real number $P(E)$ to each event $E \subseteq \Omega$ is called a probability function on the set of subsets of Ω if it satisfies the following conditions:

1. $0 \leq P(\{e_i\}) \leq 1$ for $1 \leq i \leq n$.
2. $P(\{e_1\}) + P(\{e_2\}) + \dots + P(\{e_n\}) = 1$
3. For each $E = \{e_{i_1}, e_{i_2}, \dots, e_{i_k}\}$ that is not an elementary event

$$P(E) = P(\{e_{i_1}\}) + P(\{e_{i_2}\}) + \dots + P(\{e_{i_k}\})$$

The pair (Ω, P) is called a probability space.

Theorem 2.1.1 follows from the conditions of probability given earlier that are the basis of the set-theoretic definition of probability.

Theorem 2.1.1 ([30][p.5])**.** Let (Ω, P) be a probability space. Then

1. $P(\Omega) = 1$
2. $0 \leq P(E) \leq 1$ For every $E \subseteq \Omega$.
3. For E and $F \subseteq \Omega$ such that $E \cap F = \varnothing$,

$$P(E \cup F) = P(E) + P(F)$$

With the basic definitions of the foundations of probability theory more sophisticated concepts can be introduced.

2.1.2 Conditional probability

One of the most important concepts in probabilistic graphical models is the concept of conditional probability.

Definition 2.1.2 ([30][p.6])**.** Let E and F be events such that $P(F) \neq 0$. Then the conditional probability of E given F, denoted as $P(E|F)$, is given by

$$P(E|F) = \frac{P(E \cap F)}{P(F)}$$

In the case that the knowledge about an event F does not influence the probability of an event E, these two events are called independent. Formally:

Definition 2.1.3 ([30][p.6])**.** Two events E and F are independent if one of the following holds:

1. $P(E|F) = P(E)$ and $P(E) \neq 0$, $P(F) \neq 0$
2. $P(E) = 0$ or $P(F) = 0$

In a different setting two events E and F might only be independent given the knowledge about some other event C, denoted as $E \perp F|C$. This is called conditional independence and one of the key ideas used by the models presented in this work.

Definition 2.1.4 ([30][p.7])**.** Two events E and F are conditionally independent given C if $P(C) \neq 0$ and one of the following holds:

1. $P(E|F \cap C) = P(E|C)$ and $P(E|C) \neq 0$, $P(F|C) \neq 0$
2. $P(E|C) = 0$ or $P(F|C) = 0$

A set of events E is called exhaustive if $\bigcup_i E_i = \Omega$. The events E_i are called mutually exclusive if $E_i \cap E_j = \emptyset$ for $i \neq j$.

From the definition of conditional probability the very important Bayes Theorem is derived.

Theorem 2.1.2 (Bayes [30][p.9]). Given two events E and F such that $P(E) \neq 0$ and $P(F) \neq 0$

$$P(E|F) = \frac{P(F|E)P(E)}{P(F)} \tag{2.1}$$

Furthermore, given n mutually exclusive and exhaustive events $E_1, E_2, \ldots, E_n$ such that $P(E_i) \neq 0$ for all i, for $1 \leq i \leq n$

$$P(E_i|F) = \frac{P(F|E_i)P(E_i)}{P(F|E_1)P(E_1) + P(F|E_2)P(E_2) + \ldots + P(F|E_n)P(E_n)} \tag{2.2}$$

where $P(E)$ is called the prior, $P(E|F)$ the posterior and $P(F|E)$ the likelihood of E[37][p.713]. In order to introduce probabilistic graphical models one additional concept, the concept of random variables, is needed.

2.1.3 Random variables

In order to give a mathematical description of the outcomes of experiments Random Variables are introduced. Given a probability space (Ω, P), a random variable is a function on Ω that assigns a distinct value to each element in the sample space. For a random variable X, $X = x$ is an event description which represents the event where the value of X is $x \in \Omega_X$. $X = x \wedge Y = y$ represents the case that the value of X is x and the value of Y is y. This is often abbreviated as $X = x, Y = y$. $\mathbf{P}(X)$ is the probability distribution of the random variable X and $\mathbf{P}$ assigns to each value of X a probability. Given two random variables X and Y, that are defined on the sample spaces Ω_X and Ω_Y, the joint probability distribution $\mathbf{P}(X, Y)$ is a function that assigns a probability to each event that is defined in terms of X and Y. Given the joint probability distribution $\mathbf{P}(X, Y)$, then $\mathbf{P}(X)$ can be computed as

$$\mathbf{P}(X) = \sum_{y \in \Omega_Y} P(X, Y = y) \tag{2.3}$$

This algorithm is called marginalization and the operation in 2.3 is often referred to as "summing Y out" of the distribution. In this case the distribution $\mathbf{P}(X)$ is called a marginal probability distribution.

The given insights and definitions allow the introduction of a graphical model that encodes relationships between different random variables in a concise way.

2.2 Markov networks

2.2.1 Introduction to probabilistic graphical models

Using graph structures to model the interactions of variables in a multidimensional distribution dates back to the early 20th century where it was used in physics to model the interaction of particles [15]. The study of interactions of variables was introduced to statistics with the study of contingency tables in 1935 [4]. In computer science the need to have systems that can handle uncertainty was first encountered by scientists trying to build expert systems. Early models were developed in the 60s and 70s [8; 17]. Despite successful demonstrations of the abilities of the systems developed, the research focus shifted to rule based systems because of the belief that it is best to model artificial intelligence by copying human intelligence. It was widely accepted that human intelligence would not take probabilities into account since humans do not calculate while making decisions. Probabilistic approaches reclaimed their acceptance with the invention of the Bayesian Network Framework [33] which encodes independence assumptions of random variables in a directed graphical model. Moreover the application of this framework lead to very successful systems that had less strong assumptions than the early probabilistic models. Pathfinder [19] is one of the most famous systems, where a Bayesian network is used for diagnosis of pathology samples. Undirected representation became popular because of cyclicity constraints in the directed approach as well as the ability to model symmetric influences of variables in a more compact way.

2.2.2 Introduction to Markov Networks

A Markov Network (MN) or Markov Random Field is an undirected graphical model of the joint distribution of a set of random variables. As a means to reduce the effort necessary to model the joint distribution it encodes independence assumptions of the variables.

2.2.3 Formal definition

Definition 2.2.1 ([42]). A vector X_V with random variables as elements, indexed by vertex set V, is a Markov Network over an undirected graph $G = (V, E)$, with a set of edges E, if and only if each random variable X_v, conditioned on its neighbors, is independent of all other elements of X_v with $E \subseteq V \times V$ and $v_1, v_2 \in V$, $(v_1, v_2) \in E \Rightarrow (v_2, v_1) \in E$:

$$(\forall v \in V) : X_v \perp \{X_u : u \neq v, (u, v) \notin G\} | \{X_u : (v, u) \in G\}$$

Following Definition 2.2.1, a MN encodes conditional independence assumptions and these assumptions are called the Markov assumptions. However the definition does not state anything about how a distribution is defined over the network. To make that connection the result of a famous theorem in MNs is needed.

Theorem 2.2.1 (Hammersly-Clifford [35])**.** A positive probability distribution $\mathbf{P}$ over $\mathbf{X} = \{X_i, i \in V\}$ satisfies the global Markov assumption for a graph $G = (V, E)$ if and only if $\mathbf{P}(\mathbf{X})$ factorizes according to the set of cliques $\mathcal{C}$ in G, , i.e. can be represented as a proportional product of the cliques in G.

$$\mathbf{P}(\mathbf{X}) \propto \prod_{C \in \mathcal{C}} \phi_C(\mathbf{X}_C) \tag{2.4}$$

where ϕ_C is a function that depends only on the variables $\mathbf{X}_C = \{X_{v_1}, \ldots, X_{v_n}\}$ and $C = \{v_1, \ldots, v_n\}$.

Less formally theorem 2.2.1 states that if a distribution $\mathbf{P}$ is positive and the graph encodes conditional independencies then the distribution is the product of Clique Potentials or Potential Functions over the cliques of the graph G. Clique potentials are functions that assign each state of the clique a value where a state is an assignment of a value to each variable in the clique. The distribution $\mathbf{P}$ is also referred to as the Gibbs distribution. The benefit of this representation lies in the number of parameters that need to be specified to represent a distribution. This number is the same as for the full joint distribution in the worst case but in many cases the number of parameters that need to be specified is much smaller.

The values captured in clique potentials can in the discrete case be represented as matrices or tensors where each configuration of values represents a state of the clique. As a consequence a definition of the product in 2.4 is needed in order to multiply the distributions:

Definition 2.2.2 ([24][p.107])**.** Let $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$ be three disjoint sets of variables, and let $\phi_1(\mathbf{X}, \mathbf{Y})$ and $\phi_2(\mathbf{X}, \mathbf{Z})$ be two clique potentials, called factors. The factor product $\phi_1 \times \phi_2$ is defined to be a factor $\psi :$ Val($\mathbf{X}$,$\mathbf{Y}$,$\mathbf{Z}$) $\rightarrow \mathbb{R}$ as follows:

$$\psi(\mathbf{X}, \mathbf{Y}, \mathbf{Z}) = \phi_1(\mathbf{X}, \mathbf{Y}) \cdot \phi_2(\mathbf{Y}, \mathbf{Z}) \tag{2.5}$$

In other words the product of two distributions with two variables for each distribution and one common variable, is a new distribution that contains all three variables. The values of the resulting distribution are computed by multiplying the values of the two original distributions where the truth values of the common variables overlap. An example of operation 2.5 is provided in Table 2.1.

$MC(J)$	$MC(B)$	Φ
True	True	2
True	False	1
False	True	1
False	False	1

(a) The Jackie,Bobby clique

$MC(J)$	$MC(C)$	Φ
True	True	1
True	False	2
False	True	2
False	False	1

(b) The Jackie, Charlie clique

$MC(J)$	$MC(B)$	$MC(C)$	Φ
True	True	True	2
True	True	False	2
True	False	True	1
True	False	False	2
False	True	True	2
False	True	False	1
False	False	True	2
False	False	False	1

(c) The resulting distribution

Table 2.1: Two distributions are multiplied using the factor product from definition 2.2.2.

Values of the clique potentials in a MN can be greater than 1. Consequently the sum of all states does not add up to 1, and the distribution needs to be normalized by the sum of the product of all states of the distribution.

$$\begin{aligned} P(\mathbf{X}) &= \tfrac{1}{Z} \prod_{C \in \mathcal{C}} \phi_C(\mathbf{X}_C) \\ Z &= \sum_{X \in \mathcal{X}} \prod_{C \in \mathcal{C}} \phi_C(X_C) \end{aligned} \tag{2.6}$$

with $\mathbf{X}$ being the set of all variables in the network and $\mathcal{X}$ being the set of all possible assignments of values to the network's variables. The normalizing function Z is called the Partition Function. As the definition of the partition function indicates, its computation is intractable because the number of assignments in a network grows exponentially in the number of variables. However, it can be done using sampling methods as described in [12; 21; 29].

For cliques of small size this representation is fine, however if the cliques have too many nodes then the representation becomes unclear for the designer of the network because the number of assignments that need to be done becomes too big. To overcome this problem there needs to be a representation that can exploit the structure of the potentials.

2.2.4 Log-Linear Model

In most cases the joint distribution over a MN shows some structure. A distribution often exhibits a structure where only some of the values in the distribution are of importance. Hence the distribution can be modeled by describing only the features that are relevant. In order to have to formulate only the features of interest that are entailed in the potentials of the distribution, the distribution 2.6 is transformed into a Log-Linear representation:

$$P(\mathbf{X}) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \phi_C(\mathbf{X}_C) \tag{2.7}$$

$$P(\mathbf{X}) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \exp(\ln(\phi_C(\mathbf{X}_C)) \tag{2.8}$$

$$P(\mathbf{X}) = \frac{1}{Z} \exp(\sum_{C \in \mathcal{C}} \ln(\phi_C(\mathbf{X}_C))) \tag{2.9}$$

For

$$\ln(\phi_C(\mathbf{X}_C)) = w_C \cdot f_c(\mathbf{X}) \tag{2.10}$$

the log linear model is formulated as

$$P(\mathbf{X}) = \tfrac{1}{Z} \exp(\sum_{C \in \mathcal{C}} w_C \cdot f_c(\mathbf{X})) \tag{2.11}$$

The clique potentials are transformed into one or many Feature Functions $f(x)_C$ with weights w_C. On the one end of the extremes there is one feature for each state of the clique or on the other end one feature compactly represent many states of one clique. A feature is defined as follows

Definition 2.2.3 ([24][p.125])**.** A feature $f(\mathbf{D})$ is a function over a set of variables $\mathbf{D}$ from $\mathbf{D}$ to $\mathbb{R}$.

The definition of a feature is very general and allows for a wide variety of different functions that can be used which makes it possible to represent complex structures in the distribution as a feature.

In order to build efficient learning and inference techniques an exact definition of the independence assumptions that hold in a MN is essential.

2.2.5 Independence in Markov Networks

The problem of independence in a Markov Network is the problem to determine if two variables are independent of each other just by the structure of the graph. The concept of independence in MNs is intuitive and easy to understand after the concept of an active path in a graph structure is introduced.

Definition 2.2.4 ([24][p.114])**.** Let $\mathcal{H}$ be a graph structure of a Markov Network, and let $X_1 - \ldots - X_k$ be a path in $\mathcal{H}$. Let $\mathbf{Z} \subseteq \mathbf{X}$ be a set of observed variables. The path $X_1 - \ldots - X_k$ is an active path given $\mathbf{Z}$ if none of the $X_i's$, $i = 1, \ldots, k$, is in $\mathbf{Z}$.

The concept of the active path is used to define the independence in MNs.

Definition 2.2.5 ([24][p.115])**.** A set of nodes $\mathbf{Z}$ separates $\mathbf{X}$ and $\mathbf{Y}$ in $\mathcal{H}$ denoted $sep_{\mathcal{H}}(\mathbf{X};\mathbf{Y}|\mathbf{Z})$, if there is no active path between any node $X \in \mathbf{X}$ and $Y \in \mathbf{Y}$ given $\mathbf{Z}$. The global independencies associated with $\mathcal{H}$ are defined to be:

$$\mathcal{I}(\mathcal{H}) = \{(\mathbf{X} \perp \mathbf{Y}|\mathbf{Z}) : sep_{\mathcal{H}}(\mathbf{X};\mathbf{Y}|\mathbf{Z})\} \tag{2.12}$$

This is called the global independence criterion because the criterion characterizes the entire set of independencies induced by the network structure. The global independence in MN is sound, i.e. every distribution that factorizes over a Markov Network satisfies the independence assertions implied by separation. However the global independence is not complete [24][p.115-117], i.e. it is not the case that every pair of nodes $\mathbf{X}$ and $\mathbf{Y}$ that are not separated in $\mathcal{H}$ are dependent in every distribution which factorizes over $\mathcal{H}$. For completeness only a weaker version holds:

Theorem 2.2.2 ([24][p.117])**.** Let $\mathcal{H}$ be a graph structure of a Markov Network. If $\boldsymbol{X}$ and $\boldsymbol{Y}$ are not separated given $\boldsymbol{Z}$ in $\mathcal{H}$, then $\boldsymbol{X}$ and $\boldsymbol{Y}$ are dependent given $\boldsymbol{Z}$ in some distribution P that factorizes over $\mathcal{H}$.

See [24][p.117] for the proof. This type of independence is much simpler to verify than the independence in Bayesian Networks (BNs) where d-separation asks for more elaborate algorithms(e.g. The Bayes-Ball algorithm [39]). Additionally to the global independence there are local independence assumptions.

Local Markov Assumptions

In MNs there are two local Markov independence assumptions:

Definition 2.2.6 ([24][p.118])**.** Let $\mathcal{H}$ be a graph structure of a Markov network. The pairwise independencies associated with $\mathcal{H}$ are defined to be:

$$\mathcal{I}_p(\mathcal{H}) = \{(X \perp Y | \{\mathbf{X} \setminus \{X, Y\}\}) : X - Y \notin \mathcal{H}\} \tag{2.13}$$

for $\mathbf{X}$ representing the set of all variables in the network.

The type of independence defined in 2.2.6 is called pairwise independence. One very crucial concept of independence is the Markov Blanket of a node. A node is independent of the rest of the MN given its Markov Blanket.

Definition 2.2.7. [24][p.118] For a given graph structure of a Markov Network $\mathcal{H}$ the Markov Blanket of X in $\mathcal{H}$, denoted $MB_{\mathcal{H}}(X)$, is defined to be the neighbors of a node X in $\mathcal{H}$. The local independencies associated with $\mathcal{H}$ are defined to be

$$\mathcal{I}_l(\mathcal{H}) = \{(X \perp \{\mathbf{X} \setminus \{\{X\}, MB_{\mathcal{H}}(X)\}\}) : X \in \mathbf{X}\} \tag{2.14}$$

An example is now used to illustrate the concepts developed so far.

2.2.6 An example

The following example has been introduced by Koller and Friedmann [24] and will be used throughout this work.

Example 2.2.1. Prof. Huth gives a task in class to his students. Unfortunately while doing so he makes a small mistake so that some students have a misconception. Some figure out the mistake by themselves or asking the professor after class.

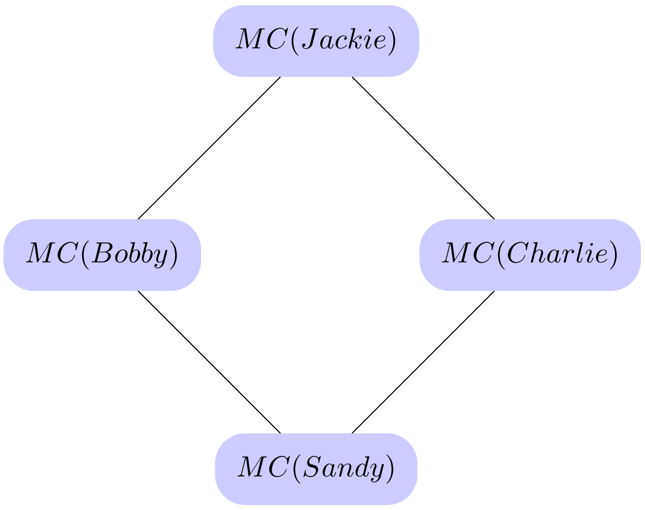

Figure 2.1: The Misconception Example as a Markov Network.

The students form small study groups to work together on the task. The students who figure out the problem may or may not share their insights with the other students. The groups of students in the case of four students in this example are $\{Bobby,\ Jackie\}$, $\{Charlie,\ Sandy\}$, $\{Sandy,\ Bobby\}$ and $\{Jackie,\ Charlie\}$. Hence the following independencies must hold in the model:

- $\{MC(Bobby)\} \perp \{MC(Charlie)\}|\{MC(Jackie), MC(Sandy)\}$
- $\{(MC(Jackie)\} \perp \{MC(Sandy)\}|\{MC(Bobby), MC(Charlie)\}$

where $MC(X)$, $X \in \{Bobby, Charlie, Jackie, Sandy\}$ models the probability that a student X has the misconception.

The example 2.2.6 represented as a MN is given in Figure 2.1 and an arbitrary distribution for the clique $\{MC(Jackie),\ MC(Bobby)\}$ is given in Table 2.2. The entries in Table 2.2 convey that it is more probable that Bobby and Jackie both have the misconception.

A more compact representation of Table 2.2 is expressed with a feature function in a log-linear model. One possible feature function for the given example is the logical AND function:

$$f(x) = \begin{cases} 1: (MC(Bobby) \wedge MC(Jackie)) = true \\ 0: otherwise \end{cases} \tag{2.15}$$

with weight $w = \ln(2)$. This yields exactly the same representation as shown in Table 2.2.

For the example the following independencies hold:

$MC(Jackie)$	$MC(Bobby)$	$\phi(MC(Jackie), MC(Bobby))$
True	True	2
True	False	1
False	True	1
False	False	1

Table 2.2: A sample for one clique of the misconception example. The potential function ϕ captures that the state where Bobby and Jackie both have the misconception is more likely than the situation where one or none of them have a misconception.

- $\{MC(Jackie)\} \perp \{MC(Sandy)\} | \{MC(Charlie), MC(Bobby)\}$
- $\{MC(Charlie)\} \perp \{MC(Bobby)\} | \{MC(Jackie), MC(Sandy)\}$

The independencies that hold in this network are a motivation to use undirected models since the independencies cannot be modeled compactly in this fashion using a BN[24][p.83].

2.3 Inference Techniques

2.3.1 Introduction

Probabilistic inference in Bayesian Networks as well as in Markov Networks is generally NP-hard or harder, meaning there are no algorithms that can solve the problem in polynomial time complexity [13][p.24]. This raises the problem of how to efficiently compute query answers to these types of networks. Different algorithms have been proposed for approximate and exact inference. Exact methods include Belief Propagation [22], the variable elimination algorithm and variants such as the forward-backward algorithm that performs inference in Hidden Markov Models [34] or the message passing algorithm for singly connected graphs [22]. Approximate inference techniques include sampling techniques like importance sampling [40], Markov Chain Monte Carlo [32] or message passing algorithms like loopy belief propagation. (see [28] for a comprehensive analysis on the latter) Since the exact inference is NP-hard the exact methods only work efficiently on special subsets of possible networks[13][p.25]. In the following first the problem of probabilistic inference and the complexity is presented. Following is the Belief Propagation algorithm which is exact in singly connected networks and most of the time converges in other networks.

2.3.2 Probabilistic Inference

One of the main tasks when using a joint probability distribution over multiple random variables is to answer queries of interest [24][p. 26]. The two most common query types include

- the conditional probability query
- the maximum a posteriori (MAP) query

Conditional Probability Query

The conditional probability query consists of two parts:

- a subset $\mathbf{E}$ of random variables in the network and an instantiation e to these variables. This is called the evidence
- a subset $\mathbf{Y}$ of random variables in the network. These are called the query variables.

The task is to compute

$$P(\mathbf{Y}|\mathbf{E} = e) \tag{2.16}$$

that is, the posterior probability distribution over the values y of $\mathbf{Y}$ conditioned on the fact that $\mathbf{E} = e$.

Maximum a Posteriori Query

The second type of queries is the task to find a high-probability joint assignment to some subset of random variables $\mathbf{X}$ given evidence e in the network. From this follows the task to compute

$$MAP(\mathbf{X}|e) = \arg\max_{\mathbf{X}} P(\mathbf{X}|e) \tag{2.17}$$

Complexity of Probabilistic Inference

In order to provide a complete analyses of the complexity of exact and approximate probabilistic inference in MNs an introduction to complexity theory is needed which is out of the scope of this work. A good introduction is found in [24][A. 3.4]. The result of an analyses of the complexity of exact and approximate probabilistic inference in MN yields that both are NP-hard. However, the complexity describes only worst cases and in many cases there exist algorithms that perform good enough for the task.

2.3.3 Belief Propagation

The belief propagation algorithm has been introduced by Pearl [31] and is a message passing algorithm. The algorithm generalizes the Bayes-likelihood updating rule 2.1.2 with the aim to efficiently and asynchronously propagate the impact of new evidence and beliefs through a complex network.

Subsequently, the details of a generalization of Pearl's algorithm are presented, the Sum-product algorithm [25].

The Sum Product Algorithm

The Sum product algorithm generalizes many existing algorithms developed in artificial intelligence, signal processing and digital communication [25]. In the following the algorithm is described in a general fashion to highlight the way it works.

The joint probability distribution factorizes over the cliques of a MN as given in 2.2.1. This factorization can be represented using a factor graph. A factor graph is a different representation of a mathematical relation between variables and local functions.

Definition 2.3.1 ([25])**.** A factor graph is a bipartite graph that expresses the structure of the factorization in 2.2.1. A factor graph has a variable node for each variable X_i, a factor node for each local function ϕ_i, and an edge connecting variable node X_i to factor node ϕ_i if and only if X_i is an argument of ϕ_i.

Figure 4.1 shows the factor graph for the MN given in 2.1 where the ϕ represents the clique potentials, and the nodes are random variables from the MN. A factor graph is used to send messages along its edges to perform probabilistic inference such as $P(MC(Jackie) = true)$. Although the algorithm is performed on an undirected graph, the algorithm assumes a tree with changing parent-children relationship depending on the edge over which messages are received. This means that at some point a node can be parent and in the next step can be child, depending over which edge a message is send.

1. At the start each node is considered a leaf in a tree.
2. Each variable leaf sends an identity message to all its parents that specifies the a priori knowledge stored in the function that creates them. Each factor leaf sends a description of f to its parents.
3. Each node waits until it received messages from all its children, i.e. messages over all edges except the one that is currently considered parent.

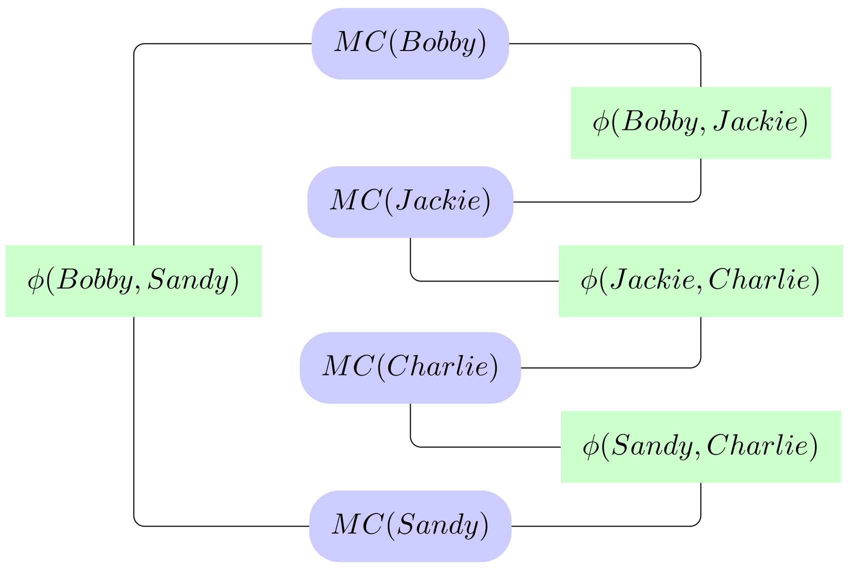

Figure 2.2: The factor graph for the misconception example.

4. When a node v received messages from all its children it computes a message to its parent:

   - if v is a function node:

   $$\mu_{f\to x}(x) = \sum_{\text{all variables but } x} \left(f(X) \prod_{y\in n(f)\setminus\{x\}} \mu_{y\to f}(y) \right) \tag{2.18}$$

   - if v is a variable node:

   $$\mu_{x\to f}(x) = \prod_{y\in n(f)\setminus\{x\}} \mu_{y\to f}(y) \tag{2.19}$$

5. The algorithm terminates, when the messages converge, i.e. the change of the messages is smaller than a defined constant ϵ, then goto 6. Otherwise, i.e. change greater than ϵ, goto 3.

6. In the termination step the belief is calculated as the product of the received messages:

   $$P(x|e) = \beta \prod_{h\in n(x)\{f\}} \mu_{h\to x}(x) \tag{2.20}$$

   with evidence e and β is a normalizing constant so that $\sum_x P(x|e) = 1$.

The formulas 2.19 and 2.18 give the name to the algorithm: The Sum Product Algorithm. In a acyclic graph only two messages have to be sent over each edge for the algorithm to terminate but in a cyclic setting, as it often occurs in MN, convergence may take more steps.

Having defined the Markov Network formalism and having shown how inference is done on this type of network, the next section introduces Logic which is the second key stone for relational models that are the subject of this work.

2.4 Logic

Logic is a powerful representation for relational data and provides sophisticated reasoning techniques. The following section follows the ideas and structure of [37][chapter 7].

2.4.1 Propositional logic

Syntax

Propositional Logic is a very simple form of logic. The syntax of propositional logic defines which sentences are allowed, where sentences arise in two forms:

- atomic sentences:
  indivisible syntactic elements, consisting of a single proposition symbol. Each such symbol stands for a proposition that either be *true* or $false$.

- complex sentences:
  constructed from simpler sentences using logical connectives and parentheses.

Sentences are either *True* or *False*. *True* and *False* are proposition symbols themselves but their value is fixed. The possible logical connectives are NEGATION($\neg$), AND($\wedge$), OR($\vee$), IMPLICATION($\Rightarrow$) and EQUIVALENCE($\Leftrightarrow$).

A literal is either an atomic sentence (positive literal) or a negated atomic sentence (negative literal). The grammar of the syntax of propositional logic is provided in table 2.4.

Semantics

The semantics of propositional logic define the rules by which the meaning of a sentence, i.e. the truth value of each formula is derived. A model, also called a world, is the set of all propositions each fixed to one truth value, i.e. for three propositions there exist $2^3 = 8$ possible models.

A	B	$\neg A$	$A \wedge B$	$A \vee B$	$A \Rightarrow B$	$A \Leftrightarrow B$
$True$	$True$	$False$	$True$	$True$	$True$	$True$
$True$	$False$	$False$	$False$	$True$	$False$	$False$
$False$	$True$	$True$	$False$	$True$	$True$	$False$
$False$	$False$	$True$	$False$	$False$	$True$	$True$

Table 2.3: The truth values of the four logical connectives in propositional logic. The precedence is highest to lowest from left to right. Parentheses can be added to change the order of execution.

$$
\begin{array}{rcl}
\textit{Sentence} & \rightarrow & \textit{Atomic Sentence} \mid \textit{Complex Sentence} \\
\textit{Atomic Sentence} & \rightarrow & \textit{True} \mid \textit{False} \mid \textit{Other proposition symbols} \\
 & & \\
\textit{Complex Sentence} & \rightarrow & (\textit{Sentence}) \mid [\textit{Sentence}] \\
 & \mid & \neg \textit{Sentece} \\
 & \mid & \textit{Sentence Connective Sentence} \\
 & & \\
\textit{Connective} & \rightarrow & \Rightarrow \mid \vee \mid \wedge \mid \Leftrightarrow
\end{array}
$$

Table 2.4: The Backus-Naur Form grammar of Propositional Logic.

The truth value of a sentence is determined by its components, the atomic sentences and the truth values of atomic sentences which are specified in the model. The truth value of the complex sentence can be determined using the truth value of the atomic sentences and the logical connectives. The truth values are obtained according to Table 2.3.

2.4.2 Decision problems in propositional logic

Two decision problems in logic are essential for the understanding of the later chapters of this work. The first, Logical entailment, is the relation $A \models B$ that holds between two sentences A and B if and only if, in every model in which A is true, B is also true.
The second, satisfiability, denotes that a sentence is true in some model. If a sentence A is true in a model M it it said that M satisfies A.[37][p.211]

2.4.3 First-Order logic

FOL generalizes propositional logic to a higher level of expressiveness. FOL has the ability to reason over sets of objects concisely whereas propositional logic does not have objects.

Syntax

FOL consists of six basic syntactic elements:

- *logical connectives* :
  as in propositional logic
- *constant symbols*:
  represent objects
- *variable symbols*:
  used to quantify over objects
- *predicate symbols*:
  represent relations
- *function symbols*:
  represent functions
- *quantifier symbols*:
  used to quantify over objects

The domain is the set of objects that the FOL deals with, it can either be finite or infinite. The grammar for the syntax of FOL is given in table 2.5.

In order to define the semantics of formulas the extended model needs to be introduced as notation. Given a model $M = \langle D, I \rangle$, a variable x and an object $o \in D$, the extended model $M_{[x \rightarrow o]}$ is a model that is identical to M, except that I is extended to interpret x as o.

Finally FOL with equality introduces the equality symbol "=". It is used to express that two terms refer to the same object.

$$
\begin{array}{rcl}
\mathit{Formula} & \rightarrow & \mathit{Primitive\ Formula} \\
& | & (\mathit{Formula\ Connective\ Formula}) \\
& | & \neg \mathit{Sentence} \\
& | & \mathit{Quantifier\ Variable\ Formula} \\
\\
\mathit{Primitive\ Formula} & \rightarrow & \mathit{Predicate}(\mathit{Term}, \ldots, \mathit{Term}) \\
\\
\mathit{Term} & \rightarrow & \mathit{Function}(\mathit{Term}, \ldots, \mathit{Term}) \\
& | & \mathit{Constant} \\
& | & \mathit{Variable} \\
\\
\mathit{Connective} & \rightarrow & \Rightarrow \mid \vee \mid \wedge \mid \Leftrightarrow \\
\mathit{Quantifier} & \rightarrow & \forall \mid \exists \\
\mathit{Constant} & \rightarrow & \mathit{a\ string\ that\ is\ not\ a\ variable\ predicate\ or\ function} \\
\mathit{Variable} & \rightarrow & \mathit{a\ string\ that\ is\ not\ a\ variable\ predicate\ or\ function} \\
\mathit{Predicate} & \rightarrow & \mathit{a\ string\ that\ is\ not\ a\ variable\ predicate\ or\ function} \\
\mathit{Function} & \rightarrow & \mathit{a\ string\ that\ is\ not\ a\ variable\ predicate\ or\ function}
\end{array}
$$

Table 2.5: The Backus-Naur Form grammar of First Order Logic.

Semantics

A first order model M is a tuple $M = \langle D, I \rangle$ where D is a non-empty domain of objects and I is an interpretation function that assigns a meaning to the syntactic elements:

- If c is a constant symbol:
  $I(c)$ is an object in the domain D.
- If f is a function symbol of arity n:
  f maps n domain objects to the domain D.
- If p is a predicate symbol of arity $n > 0$:
  $I(p)$ is a subset of D^n, i.e. a set of tuples from the domain.
- If p is a predicate symbol of arity 0:
  $I(p)$ is either $true$ or $false$.

Terms relative to a model $M = \langle D, I \rangle$ are interpreted as follow:

- If t is a constant or variable, then $t^M = I(t)$
- If t is of the form $f(t_1, \ldots, t_n)$ where f is a function symbol and the t_i are terms, then $t^M = I(f)(t_1^M, \ldots, t_n^M)$. Hence terms are interpreted as distinct objects of the domain D.

Finally, Formulas relative to a model M are interpreted as either $true$ or $false$ in the following way:

- If ϕ is a primitive formula of the form $p(t_1, \dots, t_n)$, where p is a predicate and the t_i are terms there is:

$$\phi^M = \begin{cases} true\colon if\ \langle t_1^M, \dots, t_n^M \rangle \in I(P) \\ false\colon otherwise \end{cases} \tag{2.21}$$

- If ϕ is of the form $\phi_1 \circ \phi_2$ and $\circ$ is a logical connective, $\phi^M = \phi_1 \circ \phi_2$
- If ϕ is of the form $\neg\phi_1$ and ϕ_1 is a formula, $\phi^M = \neg\phi^M$
- If ϕ if of the form $\exists x \phi_1$ where ϕ_1 is a formula:

$$\phi^M = \begin{cases} true\colon if\ there\ exists\ an\ o \in D\ such\ that\ \phi_1^{M[x \to o]} = true \\ false\colon otherwise \end{cases} \tag{2.22}$$

- If ϕ is of the form $\forall x \phi_1$ where ϕ_1 is a formula

$$\phi^M \begin{cases} true\colon if\ for\ all\ o \in D\ there\ is\ \phi_1^{M[x \to o]} = true \\ false\colon otherwise \end{cases} \tag{2.23}$$

Decision Problems in First Order Logic

The decision problem of entailment is the same as in the propositional case. That is the relation $A \models B$ if all the models of A, i.e. the models that A is true in, are also models of B. The reader should note that the task to verify this is different from the propositional case where the space of models is finite. In FOL the space can be infinite and since entailment is a statement about all models the task is usually harder.

The second decision problem, satisfiability, is also the same as in the propositional case.

2.5 Template Based Approaches

Template based representations are a compact way to represent an entire class of distributions of the same type, especially over richly structured spaces that consist of multiple objects [24][p.199]. This means that there is one distribution for each set of objects applied to a template.[24] stresses two main fields where template based representations are used.

- Temporal models
- Relational models

Temporal models deal with reasoning about the state of the world as it evolves over time. These models assume stationary processes, hence the world itself doesn't change. Some of the most important approaches include, Hidden Markov Models [34], which are a special case of Dynamic Bayesian Networks [9]. Many other representations have been proposed that generalize Hidden Markov Models or are special cases of Dynamic Bayesian Network. Relational models are the subject of Chapter 3.

2.5.1 Formalism

The key concept of Template Based approaches is the underlying template that is instantiated many times within the model. A template can be the property of a single object or even properties of tuples of objects.

The objects in the domain are divided into a set of mutually exclusive and exhaustive classes $\mathbf{Q} = Q_1, \ldots, Q_k$. Examples for such classes are "Students" and "Courses" in a university domain. Each class has attributes that are defined as follows:

Definition 2.5.1. An attribute A is a function $A(U_1, \ldots, U_k)$, whose range is some set $Val(A)$ and where each argument U_i is a typed logical variable associated with a particular class $Q[U_i]$. The tuple $U_1, \ldots, U_k$ is called the argument signature of the attribute A, and denoted as $\alpha(A)$.

The idea of having a template attribute makes it possible to instantiate a set of those to produce probability spaces with multiple random variables of the same type. In order to instantiate a set of template attributes, the set of objects for each class needs to be defined.

Definition 2.5.2 ([24][p.214])**.** Let $\mathbf{Q}$ be a set of classes, and $\mathbf{N}$ a set of attributes over $\mathbf{Q}$. An object skeleton κ specifies a fixed, finite set of objects $\mathbf{O}^\kappa[Q]$ for every $Q \in \mathbf{Q}$. Also defined is

$$\mathbf{O}^\kappa[U_1, \ldots, U_k] = \mathbf{O}^\kappa[Q[U_1]] \times \ldots \times \mathbf{O}^\kappa[Q[U_k]]. \tag{2.24}$$

By default, $\Gamma_\kappa[A] = \mathbf{O}^\kappa[\alpha(A)]$ is defined to be the set of possible assignments to the logical variables in the argument signature of A. An object skeleton may also specify a subset of legal assignments, $\Gamma_\kappa[A] \subset \mathbf{O}^\kappa[\alpha(A)]$

Ground random variables are now defined as:

Definition 2.5.3 ([24][p.215])**.** Let κ be an object skeleton over a set of classes $\mathbf{Q}$ and a set of attributes $\mathbf{N}$. Sets of ground random variables are defined as:

$$\mathbf{X}_\kappa[A] = \{A(\gamma) : \gamma \in \Gamma_\kappa[A]\} \tag{2.25}$$

$$\mathbf{X}_\kappa[\mathbf{N}] = \bigcup_{A \in \mathbf{N}} \mathbf{X}_\kappa[A] \tag{2.26}$$

In order to define a probability distribution over the ground random variables the notion of template factors is used that is defined over template attributes.

Definition 2.5.4. A template factor ξ defined over a tuple of template attributes $A_1, \ldots, A_l$, where each A_j has a range $Val(A)$. It defines a mapping from $Val(A) \times \ldots \times Val(A_l)$ to $\mathbb{R}$. Given a tuple of random variables $X_1, \ldots, X_l$, such that $Val(X_j) = Val(A_j)$ for all $j = 1, \ldots, l$, $\xi(X_1, \ldots, X)$ is defined to be the instantiated factor from X to $\mathbb{R}$.

This notation is used to introduce relational Markov networks in the next chapter.

Chapter 3

Relational Models

3.1 Overview

The attempt to represent, reason and learn in environments that are characterized by data with a complex relational and rich probabilistic structure is a very active field of research [24][Introduction p.3].

One major aspect of motivation to do research in this field is the need to model relationships between objects in a domain. This means by making a connection between two instances, a probabilistic statement about one instance is established with the knowledge about another, related instance. E.g. consider two patients that are friends. If one of them has the flue the other one is more likely to have the flue as well.

This field of research is called Statistical Relational Learning (SRL). Even if learning is not part of the current discussion the following will always refer to SRL as it is common practice in this field.

SRL research can roughly be divided into two main groups that follow two different strategies.[24]

- Frame based formalisms [1]
- First-order approaches

3.1.1 Frame Based Formalisms

The Frame Based Formalism is an approach to build systems that can range over a variety of situations whereas traditional graphical models are used for one particular situation. More precisely, systems that follow the Frame Based Formalism, model one distribution over all possible worlds for a particular set of objects. Usually this is done by using a first-order declarative

[1]In [23] this is called Knowledge Based Model Construction (KBMC) but the term is frequently used to refer to a specific approach in [45] that follows the Frame Based Formalism.

language to compactly model a template. This template is used to create a propositional graphical model, e.g. Bayesian Network or Markov Network on which inference queries can be applied [23]. Examples other than the ones presented in Sections 3.2 and 3.3 are included in [14; 20; 26]

3.1.2 First-order approaches

In first-order approaches probabilistic statements establish constraints on probabilistic models without enforcing a single, unique probability distribution [23]. The benefit of this approach is that the designer does not have to make assumptions in case no knowledge is available. First Order Logics of Probability capture probabilistic knowledge by attaching FOL formulas with probability values. Moreover formulas can be put in relation to other formulas in order to express that the knowledge expressed by one formula is more probable than the knowledge expressed by the other formula [18]. For a general overview over the different methods in that field see [27].

3.2 Relational Markov Networks

The Relational Markov Network (RMN) formalism is a representation introduced by Taskar et al. [43, 44] and extends the ideas of MNs to relational, i.e. non i.i.d. data. RMN follow a Frame Based approach where a template model is unrolled creating a MN using a relational Knowledge Base (KB).

RMNs consist of different frames, or classes as introduced in Section 2.5.1, of objects, or different entity types. Moreover an entity, more specifically an attribute of that entity, can be in relation to some other entities' attribute by participating in a relation.

3.2.1 Formalism

In recent years different notations have been proposed to describe RMNs. In the following a more recent one found in [24] is used.

Definitions

Following from the previous definitions a RMN is defined right after the introduction of template features.

Definition 3.2.1. A Template Feature $f(\lambda)$ is a strictly positive, real valued function over a set of template-level attributes as defined in 2.5.1 and logical arguments that follow a propositionalized first order logic.

Definition 3.2.2 ([24][p.229])**.** A Relational Markov Network $\mathcal{M}_{RMN}$ is defined in terms of a set Λ of template features where each $\lambda \in \Lambda$ comprises

- a real valued template feature f_λ whose arguments are $\mathbf{N}(\lambda) = \{A_1(U_1), \ldots, A_l(U_l)\}$, $\mathbf{N}$ as defined in 2.5.3.
- a weight $w_\lambda \in \mathbb{R}$

$\alpha(\lambda)$ is defined so that for all i, $U_i \subseteq \alpha(\lambda)$ where $\alpha(\lambda)$ is the signature of λ.

Using the definitions just provided a Gibbs distribution is defined over the unrolled MN for the log-linear representation.

Definition 3.2.3 ([24][p.229])**.** Given a RMN $\mathcal{M}_{RMN}$ and an object skeleton κ, a ground Gibbs distribution $\mathcal{P}_\kappa^{\mathcal{M}_{MRN}}$ is defined as follows:

- The variables in the network are $\mathbf{X}[\mathbf{N}]$
- $\mathcal{P}_\kappa^{\mathcal{M}_{MRN}}$ contains a term

$$\exp(\omega_\lambda \cdot f_\lambda(\gamma)) \tag{3.1}$$

  for each feature template $\lambda \in \Lambda$ and each assignment $\gamma \in \Gamma_\kappa[\alpha(\lambda)]$.

An edge connects two variables if they appear together in one factor in the ground MN. Important to note is that this does not mean that all predicates appearing in the description of the template features are going to be grounded variables since only the variables are grounded that match the specifications of the template features. This can considerably reduce the complexity of the resulting grounded network.

In order to connect ground variables of the network Template Features are defined at the template level that entail conditions when ground variables are connected and therefore build cliques in the ground network. Moreover to model the feature templates $f(\lambda)$ the engineer needs to know what kind of patterns he is looking for. He needs to be an expert in the field and needs to understand how the RMN works. The assignment of weights is the classical supervised learning task or done by experts.

Since RMN as defined so far only define a general concept, details of implementation are left to the user and offer a lot of freedom. The RMNs formalism does not prescribe a specific relational description language that is used to define the cliques of the model.

More elaborate introductions to the RMN formalism, with slightly different notations, are given in [13; 24; 43; 44]

For now only uncertainty with respect to the values of attributes has been considered. The next section describes how RMNs can be used to model structural uncertainty.

3.2.2 Structural Uncertainty

RMNs can be used to model structural uncertainty. [24][Section 6.6] distinguishes between two types of structural uncertainty:

- Relational Uncertainty
- Object existence Uncertainty

Relational Uncertainty

Relational Uncertainty is the uncertainty about the relations between objects in the domain which is expressed as the uncertainty about the relations between the attributes of objects in the domain. One way this could manifests itself in example 2.2.6 is in the uncertainty whether two students study together in a study group. This type of uncertainty is modeled in a concise way in RMNs using feature templates.

Changing the example from Section 2.2.6 so that study groups of size greater than two may exist, there is a feature template that models the fact that if student A studies with student B and B studies with C there is a higher probability that A studies also with C (transitivity pattern) and has the form:

$$\begin{aligned} f(S1,S2,S3) &= [\text{Study-group}(S1) = \text{Study-group}(S2)] = \mathit{true} \\ &\wedge [\text{Study-group}(S2) = \text{Study-group}(S3)] = \mathit{true} \\ &\wedge [\text{Study-group}(S1) = \text{Study-group}(S3)] = \mathit{true} \end{aligned}$$

where $Study - group(S)$ is a function symbol and encodes the study group of a particular student S. Assigning this feature template a higher weight reflects this preference [24, p.235]. Of course this changes the assumptions made earlier about the pairs of students studying together. From this example it is clear that the existence of a relationship between two objects can be encoded as another template level attribute.

The major caveat of modeling relational uncertainty using RMNs is that the model creates all possible groups. Accordingly the ground networks are dense, include lots of variables, and each clique has a large number of members. Consequently, inference and learning in the grounded network can be very hard as described in Section 2.3.2.

Object Uncertainty

Object Uncertainty addresses the problem of insufficient knowledge about the set of objects that exist in the world. An example for Object Uncertainty is the uncertainty about two points being the same in a stereo vision problem know as the Correspondence Problem or Entity Resolution Problem in social network analysis and related fields [5; 6; 38]. In the example from

Section 2.2.6 multiple instances of the same object can exist by the fact that the students are known by different writings of their name, maybe including second names or leaving them out. This leads to the problem that one student can appear multiple times in the network.

[24] proposes two methods for representing this type of uncertainty. The first introduces an additional class that represents references of objects. The members of that class symbolize the roles in the network that the objects can take, e.g. different writing of names. The instances of that object-reference class participate in a relationship with objects in the domain. Now the uncertainty is in the relation between the reference objects and the objects from the domain. As a consequence the object uncertainty problem is transformed into an instance of the relational uncertainty problem. For this approach all the objects that exist in the domain have to be known.

The second approach eliminates the genuine objects and only works with the reference instances. A "same-as(R1,R2)" predicate is introduced that the reference objects participate in, meaning a random variable expressing the belief that two instances are the same is introduced in the unrolled MN. This "same-as(R1,R2)" predicate satisfies the standard axioms of equality:

- Reflexivity
- Symmetry
- Transitivity

Additionally, the axioms of equality show that the "same-as(R1,R2)" predicate to express object uncertainty can only be used in an undirected setting. The consequence is the same in that there is a probabilistic relation. However, both suggestions have major drawbacks because they create highly peaked posteriors, i.e. only few relations are probable at all. In addition, densely connected unrolled networks are a consequence which make inference using traditional approaches hard or even intractable. A possible solution to the problem, lifted inference, is subject of Section 4.1. For a detailed discussion on structural uncertainty see [24][section 6.6].

Having defined the RMN formalism an example is used to show how this can be exploited to model relational data in a very compact way.

3.2.3 An Example

The following section extends the example that has been introduced in Section 2.2.6 to clarify the definitions that have been given in Section 3.2.1.

The model

In order to model the misconception from Section's 2.2.6 example as an RMN there has to be a definition of the classes. For example "Student" is

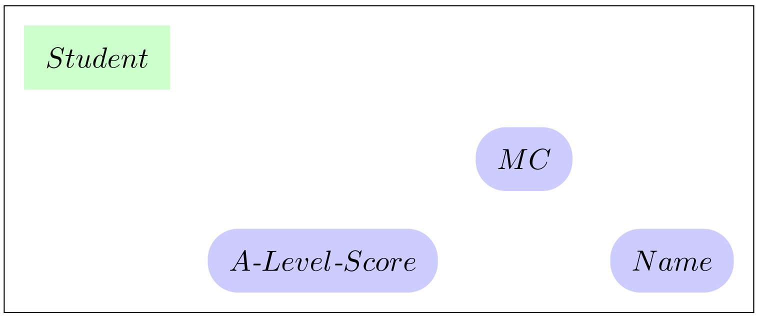

Figure 3.1: A Student class modeled in a plate like way. Three attribute descriptions that are defined for the class *Student*

Name	*A-level-score*	*MC*
Jackie	2.0	?
Bobby	1.7	?
Sandy	2.3	?
Charlie	1.0	?

(a) The student objects

*S*1	*S*2
Jackie	Bobby
Bobby	Sandy
Sandy	Charlie
Charlie	Jackie

(b) The study groups.

Table 3.1: The knowledge base of objects and relations that is used for the misconception example. All study pairs that are not mentioned are assumed to have the value $false$.

such a class. A student has the attribute "MC", "Name" and "A-Level-Score". The "MC" attribute is modeling the uncertainty whether a student has the misconception or not. "A-level score" and "Name" are observed. Intuitively the attributes "Name" and "A-level score" are not needed for the model and can be omitted. This example is supposed to show that objects are a group of attributes that do not need to have a direct relation in a specific context.

Figure 3.1 shows one entity using a symbolism known from plate models [16]. An example KB with the sample entities is given in table 3.1a and the relations of the entities in table 3.1b.

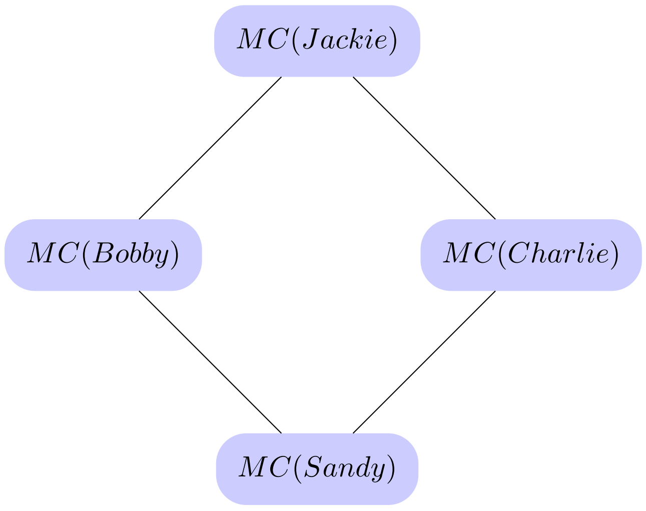

Figure 3.2: The unrolled Markov Network of the misconception example for 4 students.

The feature templates define the cliques that are created in the unrolled MN. In this example a feature template over the Variables(attributes that appear in the grounded network) "MC(S1), MC(S2)" is:

$$\begin{aligned} f(\lambda) &= \begin{cases} 1 \ \lambda \ true \\ 0 \ otherwise \end{cases} \\ \lambda =&[\text{Study-pair}(S1, S2) = true \\ &\wedge \text{MC}(S1) = \text{MC}(S2)] = true \\ &\wedge \neg(\text{S1} = \text{S2})] \\ \omega =&1 \end{aligned} \tag{3.2}$$

The feature template 3.2 expresses the tendency of students with the same level of understanding to study together. How much each feature adds to the distribution is defined by the weight assigned to the feature template and each instantiation that participates in a feature adds the same fraction. The result of the application of this feature to the KB 3.1 gives the unrolled MN in Figure 3.2. The value of the cliques are given for one sample clique in Table 3.2. As specified by the designer only the misconception variables are created. This is possible since all the study-pairs are provided in the data base. The RMN formalism requires the designer to carefully model the feature templates in order to make sure that the variables in the network have a connection.

Table 3.2 indicates that the state of Jackie and Bobby having the same state of misconception is more likely than not having the same state of

$MC(Jackie)$	$MC(Bobby)$	$\phi(MC(Jackie), MC(Bobby))$
True	True	2.7
False	True	1
True	False	1
False	False	2.7

Table 3.2: The resulting factor table.

MC(Bobby)	MC(Sandy)	MC(Jackie)	MC(Charlie)	Φ	P
true	true	true	true	53.1441	27.146%
true	true	true	false	7.29	3.723%
true	true	false	true	7.29	3.723%
true	true	false	false	7.29	3.723%
true	false	true	true	7.29	3.723%
true	false	true	false	1	0.5%
true	false	false	true	7.29	3.723%
true	false	false	false	7.29	3.723%
false	true	true	true	7.29	3.723%
false	true	true	false	7.29	3.723%
false	true	false	true	1	0.5%
false	true	false	false	7.29	3.723%
false	false	true	true	7.29	3.723%
false	false	true	false	7.29	3.723%
false	false	false	true	7.29	3.723%
false	false	false	false	53.1441	27.146%

Table 3.3: The complete unnormalized distribution with the corresponding probability of the states.

misconception, just as the feature template dictates. The structure of the tables look the same for all cliques and hence the other cliques are omitted here. The complete distribution is given in Table 3.3. To compute the probability of, e.g. Bobby and Jackie having both the misconception, the probabilities in table 3.2 have to be added, yielding:

$$\begin{aligned} P(MC(Bobby) = tr., MC(Jackie) = tr.) &= 27.147\% + 2 * 3.723\% + 0.5\% \\ &= 35.093\% \end{aligned}$$

Although no information about the state of misconception is given, the template feature implicitly makes an assumption about the misconception if two students study together. This assumption is one of the benefits of using probabilistic relational models where the knowledge about a relation indicates something about the state of the attributes.

Relational Uncertainty

The example is used to show how relational uncertainty can be modeled using RMNs.

Consider the KB previously provided in table 3.1. It is assumed that the study groups of the students are not known, leaving the relations of the students as a stochastic event. Taking the feature template 3.2 and treating the relations as non observed random binary events, the RMN yields Figure 3.3. As is evident directly from the figure, the MN is more complex having more random variables and involving now cliques of size three. Consequently the number of entries in the complete joint distribution, assuming only binary variables, has $2^{10} = 2056$ entries opposing $2^4 = 16$ entries for the previous network given in Figure 2.1.

This result captures the big problem with structural uncertainty. Inference in those networks can get intractable due to the dense structure.

Object existence uncertainty

Object uncertainty is a phenomenon in real world environments. As described in section 3.2.2 two different approaches have been suggested to tackle this challenge and these approaches reduce the problem to relational uncertainty. In the given example the students might be known by different names to the university and to different professors. The name "Bobby" for example is the american short form for "Robert". Official transcripts will most likely take the full name "Robert Lindbergh" but a list that is kept within the class might have "Bobby" and the name on the homework might be "Robert". A variable can now model the belief that "Robert" and "Bobby" refer to the real world object "Robert Lindbergh".

The second method introduces a "Same-as" variable that represents the belief that two roles represent the same "real-world" object.

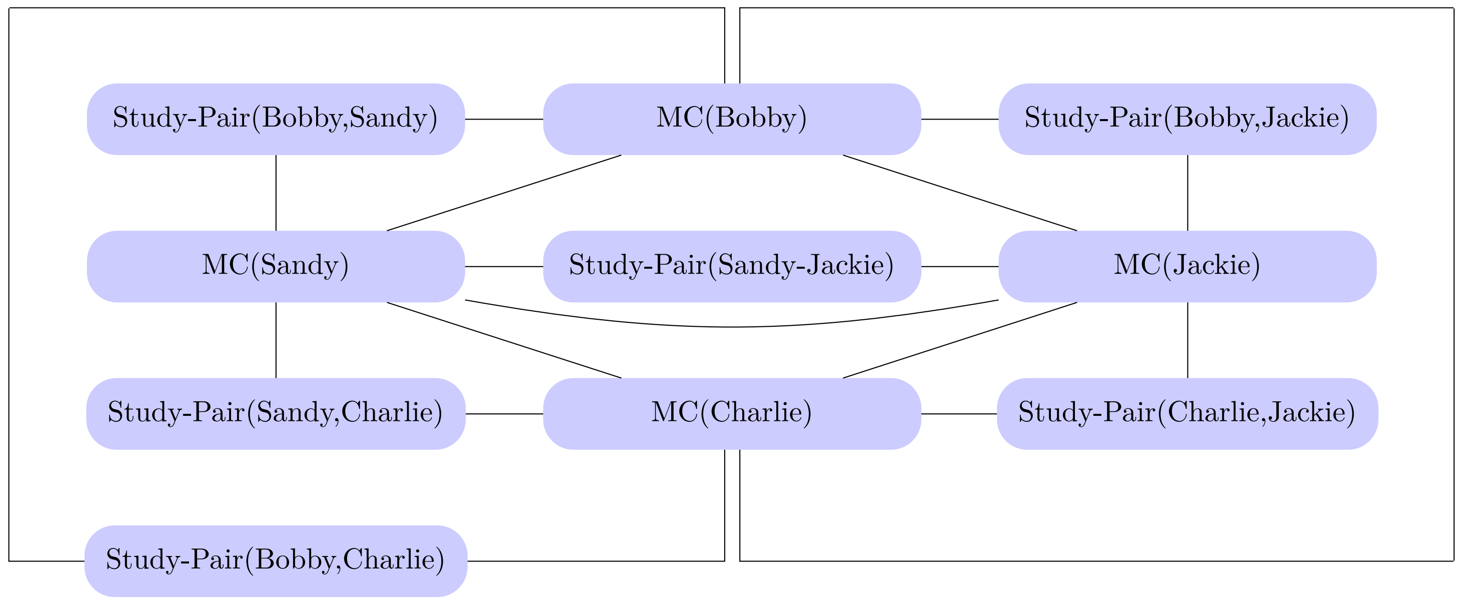

Figure 3.3: The unrolled Markov Network of the misconception example for 4 students and relational uncertainty.

3.2.4 Probabilistic Inference

Probabilistic inference is traditionally done on the unrolled MN. As a consequence the query types that need to be answered are the same as defined in Section 2.3.2. Moreover all of the existing algorithms can be used, approximate ones as well as exact ones. One has to keep in mind though that the networks resulting from templates usually are very densely connected and very big. Doing exact probabilistic inference in these networks can be very expensive. Consequently only approximate algorithms are practical.

Research tries to find a way to do probabilistic inference on the template level where we could range over a big class of models. This type of inference is called Lifted Inference and is further discussed in Section 4.1.

3.2.5 Discussion

Strengths

RMNs are a tool that has many strengths when modeling uncertainty over non i.i.d. data. First, the use of a declarative representation allows the separation of knowledge and reasoning common for all graphical models. This admits for separate development of algorithms and representation. Moreover the template design with a description language for the cliques allows an easy modeling of cliques and patterns in data. Furthermore the choice of using an undirected representation, a MN, enables one to model relationships without having to care about cyclicity and causality. Finally, the incorporation of structural uncertainty by adding additional variables for modeling of real world environments makes it a promising tool to reason over probabilistic object relational domains.

Weaknesses

The RMN formalism has many weaknesses that limit the usability as well as the guaranteed future of the model. First, the dense structure of the unrolled network in many situations make it a necessity to develop good lifted inference techniques. Especially in the presence of structural uncertainty. Likewise the absence of any temporal version of the model needs to be the subject of future research in order to transfer the model to different domains. At last in RMN the number of objects in the domains needs to be finite and the number of objects needs to be known in advance. Consequently the model is propositional.

3.3 Markov Logic

ML is a frame based approach to SRL suggested by Richardson and Domingos [36].

ML is a template based approach where a KB with weighted FOL formulas is used as a template for a MN. The basic idea of this approach relaxes the constraint of FOL that the violation of one formula in a world makes the world impossible. In ML such a world is less probable but not impossible. Intuitively ML defines a probability distribution over all possible worlds of the FOL-KB. Each formula is assigned a weight that expresses how much the violation of this formula changes the log-likelihood of the particular world where the formula is violated.

3.3.1 Formalism

Definition 3.3.1 ([11][p.12])**.** A Markov Logic Network (MLN) $\mathbf{L}$ is a set of pairs (F_i, w_i) where F_i is a formula in First Order Logic (FOL) and w_i is a real number. Together with a finite set of constants $\mathbf{C} = \{c_1, \dots, c_{|\mathbf{C}|}\}$, it defines a Markov Network (MN) $M_{\mathbf{L},\mathbf{C}}$ as follows:

- $M_{\mathbf{L},\mathbf{C}}$ contains one binary node for each possible grounding of each predicate appearing in $\mathbf{L}$. The value of the node is 1 if the ground predicate is *true*, 0 otherwise.
- $M_{\mathbf{L},\mathbf{C}}$ contains one feature for each possible grounding of each formula F_i in $\mathbf{L}$. The value of this feature is 1 if the ground formula is *true*, and 0 otherwise. The weight of the features is the w_i associated with F_i in $\mathbf{L}$.

Following Definition 3.3.1 we see that there is an edge between two nodes in the ground MN if and only if the two predicates that belong to the nodes appear together in at least one grounding of one formula. As a result the predicates in each ground formula form a not necessarily maximal clique in the ground network[11][p.13].

The MLN is a template for a MN, consequently the set $\mathbf{C}$ determines the exact MN. Different sets of constants lead to different resulting MNs. The MN that is created by one possible grounding is called a grounded MN.

Possible groundings are obtained as follows:

- replace each variable in the predicate with each constant in $\mathbf{C}$
- replace each function term in the predicate by the corresponding constant

An algorithm to obtain the groundings is given in [11, p.14]. The rules are deduced by three assumptions that ensure that the set of possible worlds for $(\mathbf{L}, \mathbf{C})$ is finite, and $M_{\mathbf{L},\mathbf{C}}$ represents a unique, well-defined probability distribution over those worlds. [11, p.13]

Assumption 3.3.1 (Unique names)**.** Different names refer to different objects.

Assumption 3.3.2 (Domain closure)**.** The only objects in the domain are those representable using the constant and function symbols in $(\mathbf{L}, \mathbf{C})$

Assumption 3.3.3 (Known functions)**.** For each function appearing in $\mathbf{L}$, the value of that function applied to every possible tuple of arguments is known, and is an element of $\mathbf{C}$

The probability distribution over possible worlds x specified by the ground MN $M_{\mathbf{L},\mathbf{C}}$ is given by:

$$P(X = x) = \frac{1}{Z} \exp\left(\sum_i w_i n_i\right) = \frac{1}{Z} \prod_i \phi_i \left(x_i\right)^{n_i(x)} \tag{3.3}$$

where Z is the partition function known from MN and n_i is the number of true groundings of the associated formula F_i in x, $x_{\{i\}}$ is the state of the predicates appearing in F_i. From the definition of the probability distribution follows that in ML no world can be impossible, i.e. have $P(X = x) = 0$. Moreover the definition of the probability distribution in 3.3 entails theorem 3.3.1

Theorem 3.3.1 ([11, p.16])**.** Let KB be a satisfiable KB, $\mathbf{L}$ be the MLN obtained by assigning weight w to every formula in KB, $\mathbf{C}$ be the set of constants appearing in KB, $P_w(x)$ be the probability assigned to a (set of) possible world(s) x by $M_{\mathbf{L},\mathbf{C}}$, $\mathcal{X}_{KB}$ be the set of worlds that satisfy KB, and F be an arbitrary formula in FOL. Then:

- $\forall x \in \mathcal{X}_{KB} \lim_{w\to\infty} P_w(x) = |\mathcal{X}_{KB}|^{-1}$
  $\forall x \notin \mathcal{X}_{KB} \lim_{w\to\infty} P_w(x) = 0$
- For all F, $KB \models F$ iff $\lim_{w\to\infty} P_w(F) = 1$

The theorem states that if all weights go to infinity, the distribution over all worlds that satisfy the KB is of a uniform nature and that every entailment query can be answered by checking whether the probability of the query formula is 1. This implies that for weights that go to infinity ML subsumes FOL over fixed domains.

In general:

Theorem 3.3.2 ([11, p.15])**.** Every probability distribution over discrete or finite-precision numeric variables can be represented as a MLN.

The proof is given in [11, p.15]. Following from Theorem 3.3.2, MLNs can be used to represent a wide variety of existing approaches to SRL. See Section 5.1 for the connection between RMNs and ML.

3.3.2 Structural Uncertainty

Structural uncertainty is modeled by loosening Assumption 3.3.1. This leads to a representation close to the one developed for RMNs in Section 3.2.2.

Relational uncertainty

Relational uncertainty is expressed by introducing predicates that encode the relationship between two objects. This predicate defines a distribution between any two objects that are grounded with that predicate. As a consequence the networks get dense with highly peaked posteriors.

Object existence uncertainty

Object uncertainty can be represented by introducing the equality predicate. The properties of the equality predicate are :

- Reflexivity
- Symmetry
- Transitivity

Practically this is done by expressing the properties in the MLN. Ergo create rules that reflect this behavior. This also introduces predicates that could be named "same-as(S1,S2)" that express the probability that two instances are the same [11, p.14].This is very similar to the techniques proposed in section 3.2.2 for RMNs.

3.3.3 Probabilistic inference

Probabilistic inference can be done with approximate or exact methods on the ground MN. In order to improve runtime the use of lifted algorithms as described in section 4.1 is a promising approach.

3.3.4 Example

The example used in Section 3.2.3 can be easily transferred into the ML formalism.

The logical formula that captures the same knowledge as the template feature 3.2 is given in equation 3.4.

$$\begin{aligned} f &= (\text{Study-Pair}(S1, S2) \\ &\quad \wedge [\text{MC}(S1) = \text{MC}(S2)] \\ w &= 1 \end{aligned} \tag{3.4}$$

A grounded network for the KB 3.1a, Figure 3.4, is almost the same as given for the relational uncertainty in RMNs. This is due to the fact that the "Study-Pair(S1,S2)" relation is grounded into a variable as it is done for all predicates in ML. However in the definition of the formula $\neg(S1 = S2)$ has been omitted. This allows for pairs of the form Study-Pair(Sandy,Sandy) to be created. The relations given in Table 3.1b can be included as formulas in the knowledge base. However they are only incorporated as priors and queries of the type "What is the probability that Jackie and Bobby form a study pair, given that they both have the misconception" are possible. Hence the relational uncertainty is automatically built into the formalism.

Structural uncertainty is included in the same way as it was done in Section 3.2.3.

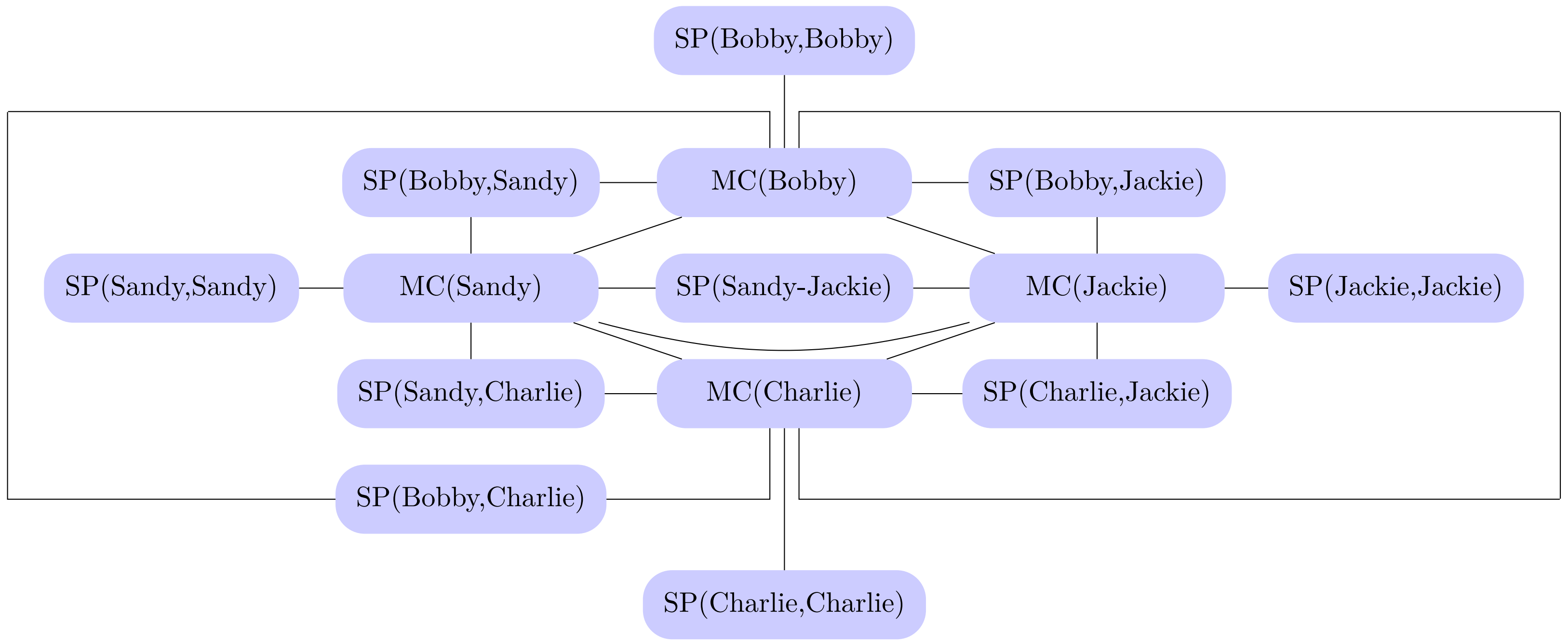

Figure 3.4: The unrolled Markov Logic Network of the misconception example for 4 students. "SP" abbreviates "Study-Pair" and "MC" abbreviates "Misconception".

Chapter 4

Improvements

4.1 Lifted Inference

Lifted algorithms are an essential development for probabilistic relational approaches to succeed in a wide range of applications. Lifted inference algorithms have the ultimate goal to run inference completely on the template level, so that no ground network needs to be instantiated in order to reduce the computational effort. A less strong goal is to make use of repeated substructures in the ground network so that as a consequence the number of computations can be reduced. These substructures exist because of the use of a template and will be exploited by the algorithm presented in the next section.

4.1.1 Lifted Belief Propagation

Lifted Belief Propagation (LBP) extends the inference algorithm described in Section 2.3.3 to exploit the structures in the resulting MN that are created because of the logical rules applied to the KB [11, p.35]. The key idea is not to instantiate all objects in the domain. In [3] an algorithm is presented that lifts inference for networks when no evidence is given. LBP for MLN extends the ideas presented in [3] to the case where evidence is presented to the network and can consequently be used to answer arbitrary queries.

The main mechanism in LBP is the identification of the nodes in the network that compute and send the same messages while performing Belief Propagation (BP) as defined in 2.18 and 2.19 to compute marginal functions. Doing these computations just once for certain structres saves computational cost

A lifted network that consists of vertices that group the vertices of the ground network as just described is defined in Section 4.1.2. Section 4.1.1 provides definitions for the nodes of the network.

Let **L** be a MLN, **C** a set of constants and *ED* an evidence database.

Definition 4.1.1 ([11, p.37])**.** A supernode is a set of groundings of a predicate that all send and receive the same messages at each step of belief propagation given **L**, **C** and *ED*. The supernodes of a predicate form a partition of its groundings.

A superfeature is a set of groundings of a clause that all send and receive the same messages at each step of belief propagation, given **L**, **C** and *ED*. The superfeatures of a clause form a partition of its groundings.

The resulting network is defined in Definition 4.1.2.

Definition 4.1.2 ([11, p.37])**.** A Lifted Network is a factor graph, as defined in Definition 2.3.1, composed of supernodes and superfeatures. The factor corresponding to a superfeature $g(x)$ is

$$exp\left(w \cdot g(x)\right)$$

where w is the weight of the corresponding first-order clause. A superfeature and a supernode have an edge between them if and only if some ground atom in the supernode appears in some ground clause in the superfeature. Each edge has a positive integer weight. A minimal lifted network is a lifted network with the smallest possible number of supernodes and superfeatures.

The regular BP algorithm is applied to the minimal network where the message from a supernode x to superfeature f is computed as follows:

$$\mu_{x\to f} = \mu_{f\to x}^{n(f,x)-1} \prod_{h\in nb(x)\setminus\{f\}} \mu_{h\to x}(x)^{n(h,x)} \tag{4.1}$$

with $nb(f)$ being all the arguments of f and $n(h,x)$ is the weight of the edge between h and x. The marginal of each supernode is given by

$$P \propto \prod_{h\in nb(x)} \mu_{h\to x}^{n(h,x)}(x) \tag{4.2}$$

Since a supernode represents a set of groundings of predicates the marginal for the supernode is the same as for all the grounded atoms in it.

The minimal lifted network is constructed in an iterative process of grounding and merging superfeatures. The outline of the algorithm is as follows:

1. Create initial supernodes by grounding all predicates and group the grounded objects by their truth values(true, false, unknown). For each predicate there are now at most three supernodes.

2. Create superfeatures by grounding clauses using all possible combinations of grounded predicates in 1). Group the groundings of the clauses by the sets they are made of(the cartesian product of the supernodes)

3. Count the number of occurrences of each atom in each position in each superfeature. Atoms with the same counts for all superfeatures are clustered into new supernodes.
   $\rightarrow$ goto 2) until convergence

Following a modified example from Table 2.2 with the change that "Jackie" and "Bobby" are considered to have the "Misconception", i.e. $MC(Jackie) = true$ and $MC(Bobby) = true$ is presented. The example shows how the algorithm checks the counts and merges the nodes to create a minimal lifted network. Table 4.1 indicates that $MC(Sandy)$ and $MC(Charlie)$ have the same counts for all superfeatures that were created in the earlier phase of the algorithm with the factor graph given in figure 4.1. As a result these two can be used to create a combined supernode in the next step of the algorithm.

The example indicates how the algorithm reduces the size of the network by checking whether two nodes send and receive the same messages [11][p.35].

	$F1$	$F2$	$F3$	$F4$	$F5$	$F6$
MC(Jackie)	1	1	–	–	1	–
MC(Sandy)	1	–	1	1	–	1
MC(Charlie)	1	–	1	1	–	1
MC(Bobby)	1	1	–	1	1	–

Table 4.1: Counts for the Misconception atom.

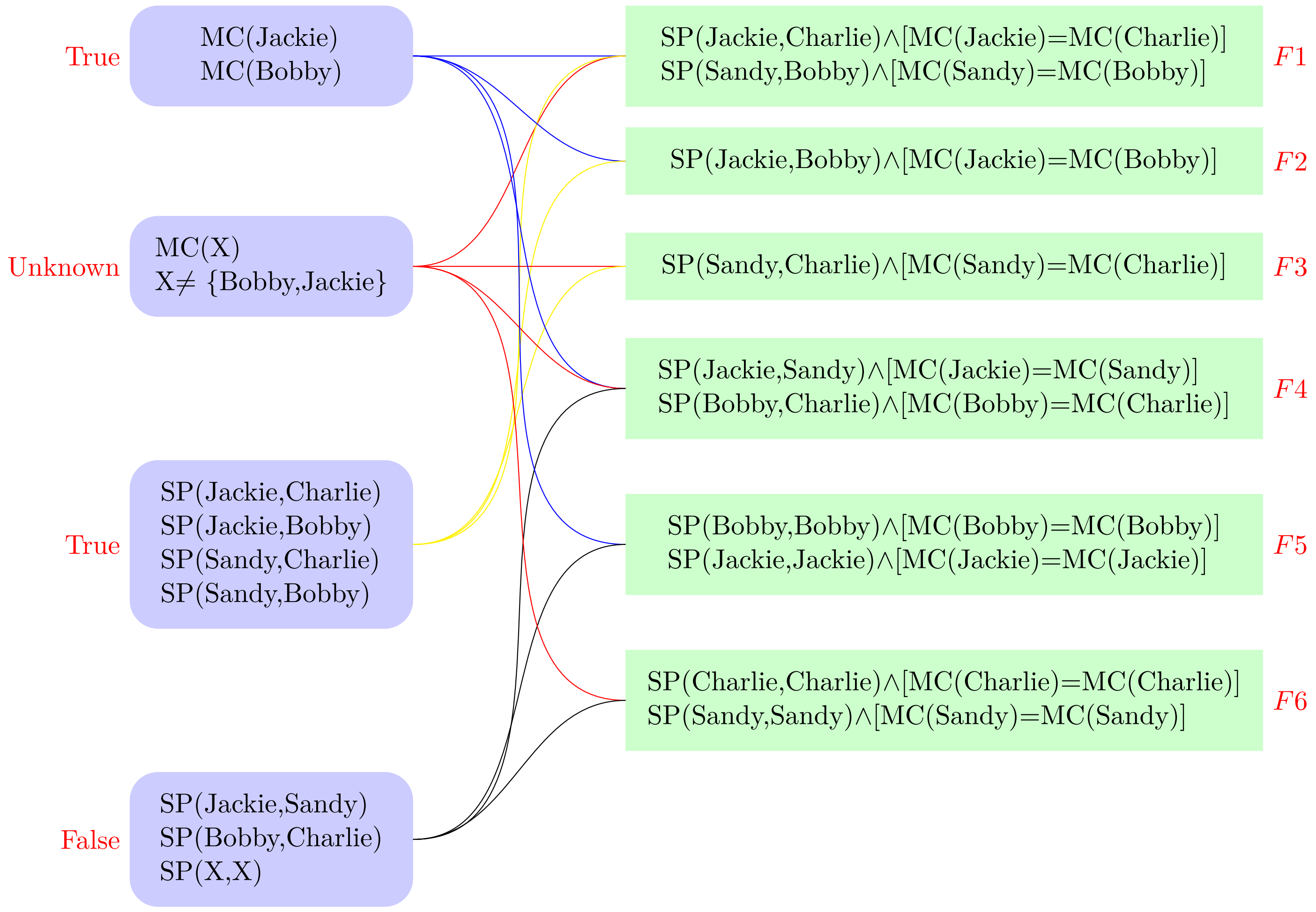
True
MC(Jackie)
MC(Bobby)
Unknown
MC(X)
X≠ {Bobby,Jackie}
True
SP(Jackie,Charlie)
SP(Jackie,Bobby)
SP(Sandy,Charlie)
SP(Sandy,Bobby)
False
SP(Jackie,Sandy)
SP(Bobby,Charlie)
SP(X,X)
SP(Jackie,Charlie)∧[MC(Jackie)=MC(Charlie)]
SP(Sandy,Bobby)∧[MC(Sandy)=MC(Bobby)]
F1
SP(Jackie,Bobby)∧[MC(Jackie)=MC(Bobby)]
F2
SP(Sandy,Charlie)∧[MC(Sandy)=MC(Charlie)]
F3
SP(Jackie,Sandy)∧[MC(Jackie)=MC(Sandy)]
SP(Bobby,Charlie)∧[MC(Bobby)=MC(Charlie)]
F4
SP(Bobby,Bobby)∧[MC(Bobby)=MC(Bobby)]
SP(Jackie,Jackie)∧[MC(Jackie)=MC(Jackie)]
F5
SP(Charlie,Charlie)∧[MC(Charlie)=MC(Charlie)]
SP(Sandy,Sandy)∧[MC(Sandy)=MC(Sandy)]
F6

Figure 4.1: The factor graph in the first stage to create a minimal factor graph for the misconception example. "MC" abbreviates "Misconception" and "SP" abbreviates "Study-Pair". The truth values of the grounded atoms are located to left of the supernodes and the names of the superfeatures are located to the right of them.

4.2 Infinite domains

The ML theory has been extended to infinite domains on a theoretical basis[11][p.75]. This is a desirable extension because it makes the model non propositional so that the complete power of FOL can be exploited. However to the knowledge of the author this extension has not yet been implemented. As a result details are not presented in this work and it is merely mentioned.

Chapter 5

Analysis

5.1 Relational Markov Network and Markov Logic Networks

MLNs can be used to represent different models and approaches to the problem of modeling uncertainty over non i.i.d. data as stated earlier. This work will focus on the relationship to the RMNs.

RMNs and MLNs are to some extend very similar approaches to the SRL problem. Both models use templates from which with the help of a relational skeleton a MN is derived.

RMNs use weighted feature templates to create cliques in the unrolled network. The same mechanism is used in ML where weighted FOL formulas are used for this purpose. The difference lies in the description languages used for the formulas and feature templates. ML dictates to use FOL which in turn create variables in the grounded network for all terms that appear in the formulas. RMNs on the other hand does not require to use any specific language. Moreover the terms that appear in the feature template do not necessarily appear in the grounded network The latter approach seems less formalized and leaves a lot of free interpretation to the developer of such systems.

The probability distribution of a RMN is given as

$$P(V_C) = \frac{1}{Z} \prod_{C \in \mathcal{C}_T} \prod_{c \in C} \exp\left(w_C \cdot f_C(V_C)\right) \tag{5.1}$$

where $\mathcal{C}_T$ is the set of clique templates and f_C is a grounded feature template with the associated weight w_C. In other words the probability distribution is given by the product over all clique templates and the product over all cliques that are created by such a template. In the special and useful case that the feature function f_C is binary function(*indicator feature*) and all terms in the feature template are grounded another interpretation is possible as well where the distribution is given as follows

$$P(X = x) = \frac{1}{Z} \prod_{C \in \mathcal{C}_T} \exp(w_c \times n_c) = \frac{1}{Z} \exp\left(\sum_{C \in \mathcal{C}_T} w_c \times n_c \right) \tag{5.2}$$

where n_c is the number of true groundings of the feature template. This representation is exactly the same as given for the probability distribution of a MLN in 3.3. Ergo, in the given case the two models are equivalent since formulas as well as the feature templates define the same cliques and factors which are going to be created in the grounded network.

As a result of the stronger formalism in ML one can conclude that ML allows to develop applications in a much more structured way. Surprisingly the lack of choice of a description language does not lead to a lack of expressiveness. Nonetheless the propositional nature of ML limits the use of FOL as a description language and doesn't make it more expressive than, for example, the SQL like description language used in some implementations of RMNs[13].

Moreover ML is in a way a much more formalized framework. This is reflected in the vast amount of research that is done in the ML community that came up with open source tools [1; 2], many extensions of the framework[11][chapter 5] as well as the active research of many different universities. On the other hand the research on RMNs seems to be less focused, i.e. not driven to the extend by its developers as it is the case for ML.

5.2 Cost and benefits of probabilistic models over object relational domains

In this section limitation and benefits of probabilistic models over object relational domains using frame based approaches are discussed.

5.2.1 Limitations

To have efficient inference techniques over probabilistic object relational domains is a key element that needs to be provided in order to let the modeling of these domains succeed in machine learning. This calls for the development of lifted inference algorithms that exploit the FOL structure of the models extensively. Although first successes are presented[7; 41] those seem not yet sufficient in order to tackle huge, complicated domains since they are in the worst case as slow as traditional algorithms and the convergence of approximate algorithms is not guaranteed.
Moreover the lack of "real" FOL as a description language for the templates, that can deal with infinite domains, limits the expressive power of the models to the propositional case.

Furthermore the number of parameters that need to be learned in those models increases potentially exponential in a domain that already was exponentially large. This asks for sophisticated algorithms whose analyses is outside the scope of this work. At last, as was pointed out by Domingos: "the biggest problem is that it is much more complex for the user to model non i.i.d. data"[10]. The designer of the system needs to understand the domain well in order to be able to model the complex relational structures that can be exploited by this modeling approach.

5.2.2 Benefits

Better predictive accuracy is one of the key goals of SRL. Results show that this goal can be met by the implementation of ML [11] as well as RMNs[14; 43].
Furthermore the necessity to model relationships between the objects in a domain leads to a better understanding of the domain of interest. This understanding is crucial while optimizing the models in order to know which information can be left out of the model so that the algorithms can perform better.

Chapter 6

Conclusion

This work introduces probabilistic graphical models. In particular two models have been presented that deal with the more complex relational data. This is achieved by the attempt of unifying First Order Logic and Probabilistic Graphical Models. Moreover it has been shown that First Order Logic is only incorporated insufficiently in Markov Logic as well as in Relational Markov Networks and used merely as a relational query language.

Despite the strong competition in the scientific community it has been shown that Relational Markov Networks and Markov Logic share many common aspects and are in fact related very closely. Nevertheless, Markov Logic as well as Relational Markov Networks have shown to perform well in real world environments.

6.1 Future Work

Future work in the field of statistical relational learning needs to address several problems that exists to date. First is the creation of more lifted algorithms that can leverage the FOL nature of the models. Furthermore it needs to be analyzed in which network structures these algorithms converge. Second the integration of infinite domains into the developer tools today and by that making real FOL a core part of the formalisms eliminating the propositional nature of the approaches. And thirdly the development of temporal relational models that can reason over time is a desirable goal.

Bibliography

[1] September 2011. URL `http://www9-old.in.tum.de/people/jain/mlns/`.

[2] September 2011. URL `http://alchemy.cs.washington.edu/`.

[3] N. F. A. Jaimovich, O. Meshi. Template based inference in symmetric relational markov random fields. *The 23rd Conference on Uncertainty in Artificial Intelligence*, 2007.

[4] M. S. Bartlett. Contingency table interactions. *Journal of the Royal Statistical Society*, Suppl. 2:248–252, 1935.

[5] P. N. Belhumeur and D. Mumford. A bayesian treatment of the stereo correspondence problem using half-occluded regions. Technical report, 1991.

[6] I. Bhattacharya and L. Getoor. Collective entity resolution in relational data. *ACM Transactions on Knowledge Discovery from Data*, 1(1):1–36, March 2007.

[7] R. D. S. Braz, E. Amir, and D. Roth. Lifted first-order probabilistic inference. In *In Proceedings of IJCAI-05, 19th International Joint Conference on Artificial Intelligence*, pages 1319–1325. Morgan Kaufmann, 2005.

[8] F. T. de Dombal, D. J. Leaper, J. R. Staniland, A. P. McCann, and J. C. Horrocks. Computer-aided diagnosis of acute abdominal pain. *British medical journal*, 2(5804):9–13, Apr. 1972. ISSN 0007-1447. doi: 10.1136/bmj.2.5804.9. URL `http://dx.doi.org/10.1136/bmj.2.5804.9`.

[9] T. Dean and K. Kanazawa. A model for reasoning about persistence and causation. *Comput. Intell.*, 5:142–150, December 1989. ISSN 0824-7935. doi: 10.1111/j.1467-8640.1989.tb00324.x. URL `http://dl.acm.org/citation.cfm?id=86605.86609`.

[10] P. Domingos. Practical statistical relational learning. Video, 2007. URL `http://videolectures.net/icml07_domingos_psr/`.

[11] P. Domingos and D. Lowd. *Markov Logic: An Interface Layer for Artificial Intelligence*. Synthesis lectures on artificial intelligence and machine learning. Morgan & Claypool, 2009. ISBN 9781598296921. URL `http://books.google.com/books?id=ijqFfoIy_TOC`.

[12] A. Gelman and X.-L. Meng. Simulating normalizing constants: From importance sampling to bridge sampling to path sampling, 1997.

[13] L. Getoor and B. Taskar. *Introduction to Statistical Relational Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2007. ISBN 0262072882.

[14] L. Getoor, N. Friedman, D. Koller, A. Pfeffer, and B. Taskar. Probabilistic relational models. In L. Getoor and B. Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press, 2007.

[15] J. Gibbs. *Elementary principles in statistical mechanics, developed with especial reference to the rational foundation of thermodynamics*. Number v. 3 in Library of American civilization. Scribner's sons, 1902. URL `http://books.google.com/books?id=2oc-AAAAIAAJ`.

[16] W. Gilks, A. Thomas, and D. Spiegelhalter. A language and program for complex bayesian modeling. *The Statistician*, 43:455–472, 1994.

[17] G. A. Gorry and G. O. Barnett. Experience with a model of sequential diagnosis. 1967.

[18] J. Y. Halpern. An analysis of first-order logics of probability. *Artificial Intelligence*, 46:311–350, 1990.

[19] D. E. Heckerman and B. N. Nathwani. An evaluation of the diagnostic accuracy. In *of Pathfinder. Computers and Biomedical Research*, pages 25–56, 1992.

[20] M. Jaeger. Relational bayesian networks. pages 266–273. Morgan Kaufmann, 1997.

[21] C. Jarzynski. Nonequilibrium Equality for Free Energy Differences. *Physical Review Letters*, 78(14):2690–2693, Apr. 1997. doi: 10.1103/PhysRevLett.78.2690. URL `http://dx.doi.org/10.1103/PhysRevLett.78.2690`.

[22] J. H. Kim and J. Pearl. A computational model for causal and diagnostic reasoning in inference systems. In *Proceedings of the Eighth international joint conference on artificial intelligence - Volume 1*, pages 190–193, San Francisco, CA, USA, 1983. Morgan Kaufmann Publishers Inc. URL `http://dl.acm.org/citation.cfm?id=1623373.1623417`.

[23] P. Klinov. *Practical reasoning in probabilistic description logic.* PhD thesis, University of Manchester, 2011.

[24] D. Koller and N. Friedmann. *Probabilistic Graphical Models: Principles and Techniques.* Number 14 in Adaptive Computation and Machine Learning. The MIT Press, Cambridge, Massachusetts, 2009.

[25] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47:498–519, 1998.

[26] K. B. Laskey. Mebn: A language for first-order bayesian knowledge bases. *Artificial Intelligence*, 172:140–178, 2008. doi: 10.1016/j.artint.2007.09.006.

[27] T. Lukasiewicz. Expressive probabilistic description logics. *Artif. Intell.*, 172:852–883, April 2008. ISSN 0004-3702. doi: 10.1016/j.artint.2007.10.017. URL `http://dl.acm.org/citation.cfm?id=1342435.1342785`.

[28] D. M. Malioutov, J. K. Johnson, and A. S. Willsky. Walk-sums and belief propagation in gaussian graphical models. *J. Mach. Learn. Res.*, 7:2031–2064, December 2006. ISSN 1532-4435. URL `http://dl.acm.org/citation.cfm?id=1248547.1248620`.

[29] R. M. Neal. Annealed importance sampling. *Statistics and Computing*, 11:125–139, April 2001. ISSN 0960-3174. doi: 10.1023/A:1008923215028. URL `http://dl.acm.org/citation.cfm?id=599243.599401`.

[30] R. E. Neapolitan. *Learning Bayesian Networks.* Prentice Hall, illustrated edition edition, Apr. 2003. ISBN 0130125342. URL `http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0130125342`.

[31] J. Pearl. Reverend Bayes on inference engines: A distributed hierarchical approach. In *Proceedings of the American Association of Artificial Intelligence National Conference on AI*, pages 133–136, Pittsburgh, PA, 1982.

[32] J. Pearl. Evidential reasoning using stochastic simulation of causal models. *Artif. Intell.*, 32:245–257, May 1987. ISSN 0004-3702. doi: 10.1016/0004-3702(87)90012-9. URL `http://dl.acm.org/citation.cfm?id=25672.25675`.

[33] J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988. ISBN 0-934613-73-7.

[34] L. Rabiner and B. Juang. An introduction to Hidden Markov models. *ASSP Magazine, IEEE*, 3(1):4–16, Apr. 2003. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1165342`.

[35] P. Ravikumar. Approximate inference, structure learning and feature estimation in markov random fields: thesis abstract. *SIGKDD Explor. Newsl.*, 10:32–33, December 2008. ISSN 1931-0145. doi: http://doi.acm.org/10.1145/1540276.1540286. URL `http://doi.acm.org/10.1145/1540276.1540286`.

[36] M. Richardson and P. Domingos. Markov logic networks. In *Machine Learning*, page 2006, 2006.

[37] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2 edition, 2003. ISBN 0137903952.

[38] D. Scharstein, R. Szeliski, and R. Zabih. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. In *Proceedings of the IEEE Workshop on Stereo and Multi-Baseline Vision (SMBV'01)*, SMBV '01, pages 131–, Washington, DC, USA, 2001. IEEE Computer Society. ISBN 0-7695-1327-1. URL `http://portal.acm.org/citation.cfm?id=580204.884062`.

[39] R. D. Shachter. Bayes-ball: The rational pastime. In *In Uncertainty in Artificial Intelligence*, pages 480–487. Morgan Kaufmann, 1998.

[40] R. D. Shachter and M. A. Peot. Simulation approaches to general probabilistic inference on belief networks. In *Proceedings of the Fifth Annual Conference on Uncertainty in Artificial Intelligence*, UAI '89, pages 221–234, Amsterdam, The Netherlands, The Netherlands, 1990. North-Holland Publishing Co. ISBN 0-444-88738-5. URL `http://dl.acm.org/citation.cfm?id=647232.719570`.

[41] P. Singla and P. Domingos. Lifted first-order belief propagation. Technical report, Department of Computer Science and Engineering University of Washington, WA 98195-2350, U.S.A., 2008.

[42] N. Srebro and D. Karger. Learning markov networks: maximum bounded tree-width graphs. In *In Proceedings of the 12th ACM-SIAM Symposium on Discrete Algorithms*, pages 392–401, 2001.

[43] B. Taskar, M. Wong, P. Abbeel, and D. Koller. Discriminative probabilistic models for relational data. Technical report, Computer Science Dept. Stanford University, Stanford, CA 94305, 2002.

[44] B. Taskar, M. Wong, P. Abbeel, and D. Koller. Link prediction in relational data. *Proceedings of Neural Information Processing Systems*, 2003.

[45] M. Wellman, J. Breese, and R. Goldman. From knowledge base to decision models. *Knowledge Engineering Review*, 7:35–53, 1992.